Home

Welcome to the official Irony Mod Manager Wiki.
This documentation provides detailed information about Irony’s features, workflows, configuration options, and advanced tools.

Irony Mod Manager is a high-performance mod manager for Paradox titles, offering deterministic mod ordering, conflict analysis (game-dependent), mod merging, and deep integration with Stellaris, EU4, HOI4, CK3, V3, Imperator, and Star Trek: Infinite.

This Wiki is organized into clear sections accessible through the sidebar.
If you are new to Irony, start with the **New User Checklist**.

□ Getting Started

If you're just beginning with Irony:

- Begin with the **New User Checklist**
- Learn how installed mods appear in Irony → **Installed Mods**
- Understand how to create and manage a Collection → **Collection Mods**
- Explore sorting, filtering, and basic UI elements → **Mod Filter**

Irony works on Windows, Linux and macOS, and supports portable operation on all platforms.

□ Key Features (Overview)

Irony offers:

- **Deterministic mod load ordering** based on game-specific rules
- **Conflict Solver** (Full for Stellaris, analysis for HOI4) → **Conflict Solver Modes**
- **Mod Merging** with multiple merge strategies → **Merge Viewer** / **Binary Merge Viewer**
- **Patch Mod auto-generation** when resolving conflicts
- **Searchable database view** for advanced conflict inspection → **Database Search**
- **External merge tool integration**
- **Custom ignore rules**, override rules and patch instructions → **Ignore Rules**
- **Platform-specific rendering and performance options**

□ Documentation Structure

Use the sidebar to navigate through:

User Workflow

- Installed Mods
- Installing New Mods
- Collection Mods

- Mod Filters

Conflict Solver & Tools

- Conflicted Objects
- Merge Viewer
- Binary Merge Viewer
- Ignore Rules
- Reset Conflicts
- Database Search
- Custom Patches

Configuration

- Options
- Keyboard Shortcuts
- Adding Irony to Steam

Reference

- FAQ
- Privacy Policy
- Credits

□ Additional Resources

For release notes, downloads, and platform-specific installation steps, refer to the GitHub Releases page.

For interactive help, tutorials, and discussions, visit the Irony Discord community.

□ Need Help?

If you encounter issues:

- Check the FAQ
- Use the Troubleshooting section (in FAQ)
- Or open a bug report on GitHub (with logs and minimal reproduction)

This Wiki is continuously maintained as Irony evolves.
Thank you for using Irony Mod Manager!

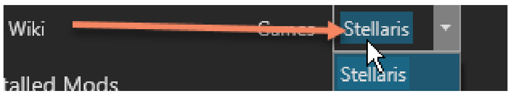
New User Checklist

Welcome! This guide shows the essential steps to get started with **Irony Mod Manager**.
It uses real screenshots of the UI to help new users understand where everything is.

Irony manages collections, mod order, merging, and (game-dependent) conflict solving.
Follow the steps below to set up your first working collection.

1. Select a Game

When Irony starts, you will see the **game selection screen**.
Choose the Paradox game you want to manage.

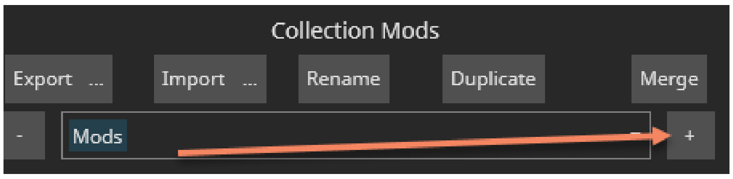


Irony will remember the selected game on next launch.

2. Create a New Collection

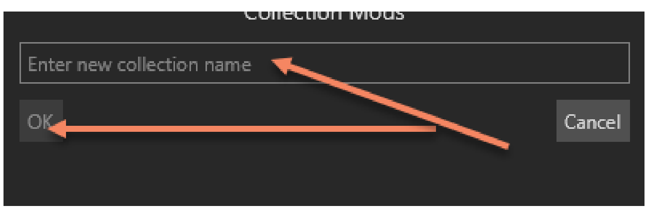
Collections define which mods are active and in which order.

Click **Create New Collection (+ button)**.



3. Enter a Collection Name

Choose any name you like — for example “My Stellaris Setup” or “CK3 Mods”.

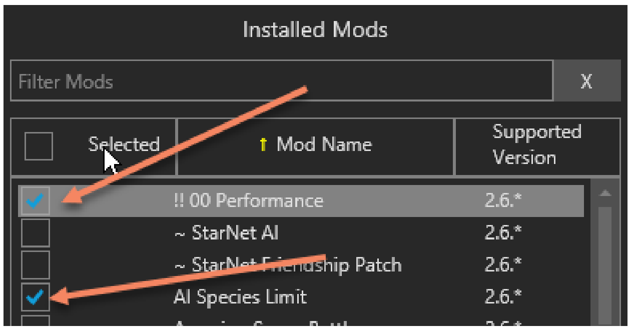


Click **OK** to create it.

4. Select Mods in the Installed Mods Area

On the bottom-left area, you will see your **Installed Mods** list
→ Installed Mods

Select the mods you want to include.



Selected	Mod Name	Supported Version
✓	!! 00 Performance	2.6.*
	~ StarNet AI	2.6.*
	~ StarNet Friendship Patch	2.6.*
✓	AI Species Limit	2.6.*

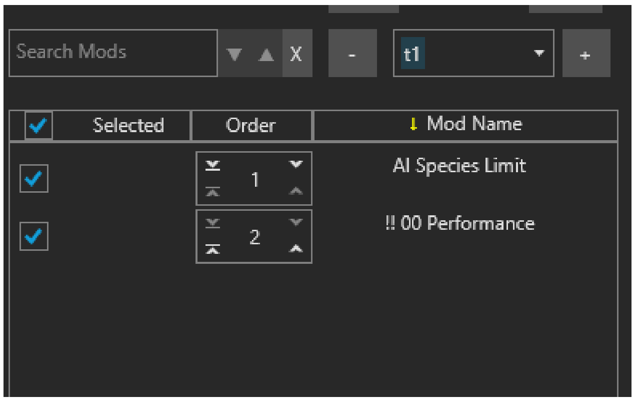
You can select multiple mods using Ctrl or Shift as needed.

5. Mods Will Appear Under “Collection Mods” (Right Side)

Selected mods appear on the **right side** of the UI, under **Collection Mods**

→ Collection Mods

This is your actual load order.



Selected	Order	Mod Name
✓	1	AI Species Limit
✓	2	!! 00 Performance

You can drag them up and down to change order.

Optional: Run the Conflict Solver

- **Stellaris** → Full conflict solver
- **HOI4** → Analysis only
- **Others** → Not available

If available, you can inspect conflicts, apply merges or patches, and clean up the patch mod by re-running the solver.

Optional: Merge Mods

You can merge mods via merge menu:

- Basic Merge
- Compress Merge

Merged mods appear as local mods and can be used in any collection.

6. Apply or Launch Game

Once everything looks good:

- Press **Apply** to write the load order
- Press **Launch Game** to start the game with your collection active

Troubleshooting

If something goes wrong:

- Use **Actions** → **Logs** to open the log folder
- Check the Troubleshooting section of the Wiki → FAQ
- If reporting an issue, **provide a minimal reproducible collection**, not your entire setup

This checklist covers the basics needed to use Irony for the first time.
Once you're comfortable, explore advanced tools like Filtering, Database Search, Ignore Rules, Merge Viewer, and external

editor integration.

Installed Mods

The **Installed Mods** panel lists all mods detected on your system.
This view does **not** decide whether a mod is enabled or disabled in the game -- it only shows what exists on disk.

Mods become active/inactive when they are added to or removed from a **Collection**:
→ See: Collection Mods

1. Overview

The Installed Mods panel allows you to:

- View all installed mods (Steam, Paradox, local)
- Filter and search using advanced syntax
- Sort mods by any column
- Move mods into or out of a Collection
- Inspect and regenerate mod descriptors
- Lock/unlock descriptors as readonly
- Rescan for newly installed mods

This panel works independently from the **Collection Mods** panel.

2. Filtering Mods

See **Filter Mods** for the full syntax reference.

Filtering supports:

- Mod name
- Remote ID
- All advanced operators and field filters

Resetting the Filter

- Click the **X** button in the textbox, or
- Delete all text.

3. Sorting Mods

Click any column header to sort by that column.

Sort order and selected column are saved automatically.

Supported sort columns include:

- Name
- Version
- Source

- Achievement compatibility
- Selection status
- Remote ID

4. Moving Mods to the Collection Mods panel

To activate a mod in-game:

1. Tick the **Selected** checkbox.
2. The mod appears in **Collection Mods**.
3. The game will load it as part of that collection.

To deactivate:

- Untick the checkbox — the mod is removed from the Collection.

4.1 Bulk Selection

You can enable/disable many mods at once:

- Click the **checkbox in the column header**
- Together with filters, this affects only **visible mods**

Example workflow:

```
Filter: source:steam && achievements:no
Click header checkbox → toggles only these mods
```

5. Saved Options

Irony automatically remembers:

- Sort column
- Sort direction
- Filter text

Settings persist instantly.

6. Determining Achievement Compatibility

Next to each mod, Irony shows an icon:

- **Broken trophy** → the mod is not achievement compatible
- **Intact trophy** → the mod is achievement compatible

This integrates with filtering (`achievements:yes/no`).

7. Context Actions (Right-Click Menu)

Right-click a mod to access advanced actions:

Open Mod File/Directory
Opens the mod folder in your file explorer.

Open Mod Url
Opens the mod's homepage in your browser.

Open in Steam
Opens the mod's Workshop page in the Steam client.

Copy Mod Url
Copies the mod homepage URL.

Check New Mods
Rescans workshop and local mod folders and generates descriptors.

Delete And Reload
Deletes descriptor for the selected mod and regenerates it.

Delete All And Reload
Deletes descriptors for all **visible** mods and regenerates them.
Works well with filters.

Lock
Sets the readonly attribute on the descriptor file of the hovered mod.

Lock All
Locks descriptors for all **visible** mods.

Unlock / Unlock All
Removes the readonly attribute from descriptor files.

8. Notes

- The Installed Mods panel does **not** control load order — Collections do.
- Selection controls activation; this panel shows only what exists on disk.
- Descriptor regeneration is safe and produces normalized files.

Installing New Mods

Irony Mod Manager does **not** install mods itself.
Mods must be installed through the game's platform (Steam, Paradox Mods, manual folders), and Irony will detect them automatically.

This page explains how mods are installed depending on the platform and how to ensure Irony sees them.

Steam Workshop Mods

To install a mod from Steam:

1. Open the mod's Steam Workshop page
2. Click **Subscribe**
3. Wait for Steam to download the mod
4. Launch the game once (optional but recommended)
5. Start Irony — the mod will appear under **Installed Mods**
   → Installed Mods

No manual steps are required.
Irony will detect the mod as soon as Steam finishes downloading it.

Paradox Mods

To install a mod from Paradox Mods:

1. Go to the Paradox Mods page for your game
2. Click **Subscribe**
3. Launch the Paradox Launcher
4. It will download the mod into the Paradox Mods directory
5. Start Irony — the mod will appear under **Installed Mods**
   → Installed Mods

Local Mods

Local mods are mods placed manually in the game's mod folder.

Examples of local mods:

- Mods downloaded manually as ZIP or folder
- Mods created using Irony's merge features
  → Collection Mods
- Mods exported from a collection
  → Collection Mods
- Mods created by the conflict solver (patch mod)
  → Conflict Solver Modes

To install a local mod:

1. Extract the mod into the game's local mod folder
2. Ensure it contains a valid `.mod` or `.json` descriptor file
3. Start Irony — the mod will appear under **Installed Mods**
   → Installed Mods

Forcing Irony to Detect Newly Installed Mods

If you installed a mod and it does not appear immediately, use:

Right Click → Check New Mods
(Available in Installed Mods context menu.)

This forces Irony to rescan:

- Steam Workshop directory
- Paradox Mods directory
- Local mod directory

Any new mods found will be added automatically.

→ See full list of actions under: Installed Mods

Descriptor Requirements

A mod must contain:

- A valid descriptor file (`.mod` or `.json`)
- A content root (folder that matches the descriptor's path)

If the descriptor is missing, corrupted, or incomplete, Irony will show a popup warning.

Summary

- Steam → Subscribe → wait → Irony detects
- Paradox Mods → Subscribe → Launcher downloads → Irony detects
- Local mods → extract + valid descriptor → Irony detects
- Use **Check New Mods** for rescanning
- Irony never installs mods — it only detects what the system provides

This page covers everything related to installing mods for Irony.

Collection Mods

The **Collection Mods** panel is where you manage the mods that will be enabled in the game.
Here you control your **active mod list**, **load order**, **exports/imports**, **merging**, and access advanced tools such as hashing and conflict solving.

A collection must be selected before editing. If the dropdown list is empty, you must create a collection first.

1. Overview

The Collection Mods section (right side of the UI) allows you to:

- Choose which mods are active in the collection
- Reorder mods (load order)
- Search and filter mods
- Merge, export, or import collections
- Apply the load order to the game
- Launch or resume the game
- Access the Conflict Solver → Conflicted Objects
- Manage patch mods
- Run hash verification for multiplayer sync

2. Collection Management (Dropdown Actions)

The dropdown above the Collection Mods list represents the **currently selected collection**.

Create Collection

Click the **+** button next to the dropdown.
Enter a name and confirm.

Delete Collection

Click the **–** button.
This permanently deletes the collection and its associated patch mod.

Duplicate Collection

Creates a new copy following Windows file naming conventions (e.g., “Collection (2)”).

Rename Collection

Click the rename button and enter a new name.
Both the collection and its patch mod (if present) are renamed.

3. Exporting Collections

Click the **Export** button to save a collection as a ZIP file.
Available export modes:

Standard Export

Exports:

1. Collection data (name, mod order)
2. Patch collection mod (if it exists)

Additional Export Options (. . . menu)

Export Order Only

Exports only the collection name and mod order.

Export Whole Collection

Exports:

- Patch mod
- Load order
- **All mods in the collection**

Useful for sharing exact setups.

Export to Paradox Launcher JSON (< 2021.10)

Exports a load order in the older Paradox Launcher JSON format.

Export to Paradox Launcher JSON (>= 2021.10)

Exports for newer Paradox Launcher versions.

Note:
Paradox Launcher exports only non-local mods. Irony follows the same rule.

4. Importing Collections

Click **Import** to load a previously exported Irony collection.

Imports:

1. Collection data
2. Patch collection mod (if any)

If a collection with the same name exists, Irony will prompt you to overwrite it.

Additional Import Options (. . . menu)

Import from Paradoxos

Imports Paradoxos load orders.

Import from the Game

Reads `dlc_load.json` and creates a new collection named **"Paradox"**.

Import from Paradox Launcher

Imports active playset from the Paradox Launcher SQLite database.

Import from Paradox Launcher Beta

Same as above, but uses the launcher's beta database.

Import from Paradox Launcher JSON

Imports a Paradox Launcher JSON export (all supported formats accepted).

5. Managing Mods Within a Collection

Adding Mods

Select mods in **Installed Mods** → Installed Mods
and add them to the collection.
They will appear in the right-side list.

Searching Mods

Search performs partial matching by mod name or remote ID.
Advanced syntax is supported:

- `term` (partial match)
- `a && b` (AND)
- `a || b` (OR)
- `--a` (NOT)
- `achievements:yes/no`
- `selected:yes/no`
- `source:steam/paradox/local`
- `version:x.y.z`

Use the up/down buttons next to the search box to navigate between matches.
(See full syntax: → Mod Filter)

Sorting Mods

Click the **mod name** column header to sort alphabetically.

Reordering Mods

Load order controls override behavior — **bottom mods override mods above them**.

Reorder using:

- Drag & drop
- Numeric up/down buttons
- Editing the position number directly

Saved Options

Irony remembers:

- Sort order
- Search text

6. Patch Mod Controls

Irony visually indicates when a **patch mod** exists.

Options available:

- Toggle patch mod on/off
- Delete patch mod fully

Patch mods are created by the Conflict Solver → Conflicted Objects

7. Conflict Solver

Clicking **Conflict Solver** opens the solver.
First-time use will prompt you to select a mode:

(Modes explained here → Conflict Solver Modes)

Important Notes:

- Changing the collection requires rerunning the solver.
- Large mod lists may take time and memory.
- Solver always prompts before repeated runs.

8. Context Actions (Right-Click Menu)

Right-click inside Collection Mods to access:

Auto Focus

Toggles automatic selection of moved items.

Copy Collection to Clipboard

Copies mod names into clipboard for sharing/debugging.

Import Mod Order From Clipboard

Reads clipboard and sets mod order accordingly.

Export Collection Hashes

Exports file hashes for all mods in the collection.

Export Game Hashes

Exports hashes for game installation files.

Verify Hashes

Validates the local files against a hash report.

Open Mod File/Directory

Opens the mod folder in your file explorer.

Open Mod URL

Opens the mod's homepage in your browser.

Open in Steam

Opens the mod's Workshop page in the Steam client.

Copy Mod URL

Copies the mod's homepage URL.

9. Apply & Launch

Apply

Writes the collection load order to the Paradox Launcher / game registry.

Note:
Changes done directly in Paradox Launcher do **not** sync back to Irony.

Launch Game

Applies the collection and runs the configured executable.
If Steam is required but not running, Irony will start Steam first.

Resume Game

Launches the game and loads the **last saved** game automatically.

Shown only when a valid `continue_game.json` exists.

10. Notes & Limitations

- Collections cannot switch mod versions (per Paradox limitations).
- Paradox Launcher changes do not modify Irony collections.
- Patch mods must be regenerated if the collection is changed.
- Importing/exporting collections overwrites patch mods when names match.

This page documents all functionalities of the Collection Mods panel.

Mod Filter

The **Filter Mods** search box allows you to quickly find mods by name, ID, or advanced criteria.
Filtering supports partial matching, logical operators, negation, and field-specific queries.

This system is powerful and flexible, allowing both simple searches and complex filters.

Filtering is available in the **Installed Mods** panel:
→ Installed Mods

Basic Filtering

Typing any text will show all mods whose name or remote ID **contains** the search term.

Examples:

- `star` → matches “Star Wars Mod”, “Starlane Improvements”
- `1234` → matches mods with remote ID containing 1234

Matching is **partial** and **case-insensitive**.

Logical Operators

AND Operator

Use `&&` to combine multiple filters:

```
my mod && source:steam
```

OR Operator

Use `||` to match either condition:

```
my mod || another mod
```

NOT Operator (Negation)

Use `--` before a term or field:

```
--my mod
source:--local
```

Negation works on **all** supported fields.

Field Filters

You can filter mods using structured keywords:

Achievements

```
achievements:yes
achievements:no
achievements:true
achievements:false
```

Selected (in the current collection)

```
selected:yes
selected:no
```

Source

```
source:steam
source:paradox
source:local
```

Supports OR syntax:

```
source:steam||paradox
```

Version

Matches version fields declared in descriptors:

```
version:2.0
version:3.2.1
version:2.0||3.0
```

(Partial matching applies.)

Combined Examples

Match a specific mod from Steam that supports achievements:

```
my mod && achievements:yes && source:steam && selected:no && version:2.0
```

OR operator example:

```
my mod || another mod && source:steam||paradox
```

Negation example:

```
--my mod && source:--local
```

Summary

Filtering in Irony allows:

- Partial name or ID searches
- Combining multiple criteria
- Logical AND / OR
- Negation
- Filtering by achievements support
- Filtering by selection state
- Filtering by source (steam / paradox / local)
- Filtering by version

This system makes managing large mod lists fast and efficient.

Conflicted Objects

The **Conflicted Objects** view shows all detected conflicts for the currently selected collection.
It presents a structured, game-logic hierarchy where you can inspect definitions, understand which mods touch the same objects, and open them in the Merge Viewer for resolution → Merge Viewer.

1. Overview

Conflicted Objects are displayed in two parts:

- A **dropdown** showing the full conflict hierarchy (folders → files → definitions)
- A **listbox** showing the definitions inside the selected file or node

Selecting a definition opens the **Merge Viewer**, where differences between mods can be examined and resolved (if the game supports full merging).
→ Merge Viewer

This view is the starting point for analyzing and fixing mod conflicts.

2. Navigating Conflicts

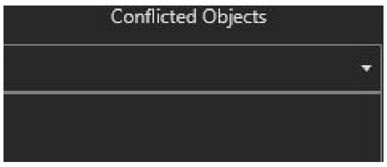
Conflict Hierarchy Dropdown

The dropdown displays all conflicted files and logic nodes Irony has detected.
Each entry expands the game's internal structure, allowing you to drill down to the exact definition that has conflicting contributors.

Example:

- `common`
- `common/ship_behaviors`
- `common/ship_behaviors/behavior_type_1.txt`

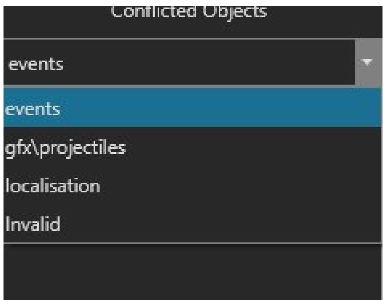
Simply select a node to load its definitions in the listbox.

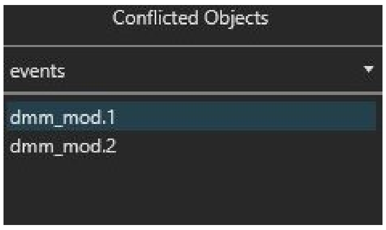


Definitions Listbox

After selecting a node, the listbox shows all definitions inside that file that have conflicts.

Selecting a definition loads it into the Merge Viewer → Merge Viewer





3. Reset Indicators

Irony automatically **resets conflicts** when it detects that mod files have changed
(for example, a mod author updated a file after you previously resolved or ignored a conflict).

Reset indicators help highlight these cases:

- **Red border** → previously resolved conflict has changed
- **Yellow border** → previously ignored conflict has changed

These indicators appear directly in the definitions listbox.

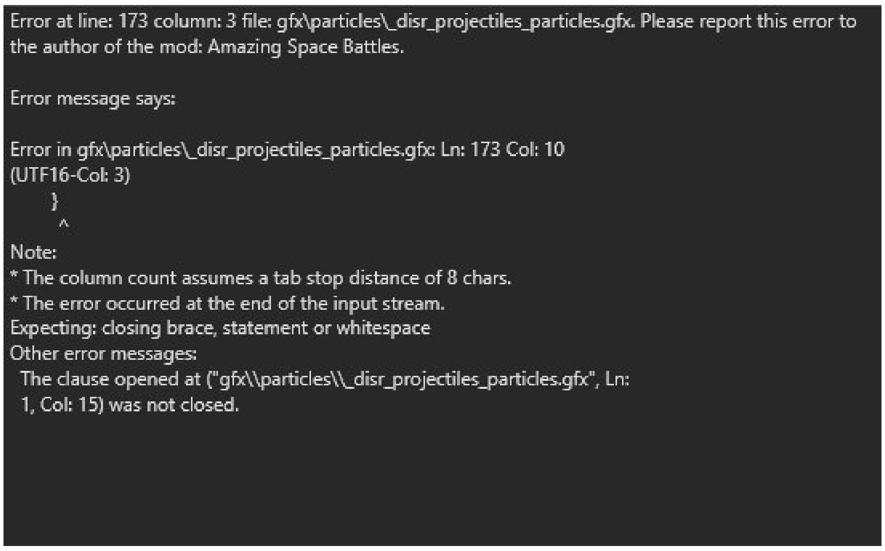
You can filter to only show reset conflicts by enabling:
Conflict Filter → **Show Only Reset Conflicts** → Conflict Filter

4. Invalid Objects

Occasionally, Irony can detect **invalid definitions**.
This analysis is powered by CWTools.

Invalid items appear in a separate dropdown section labeled **Invalid**.

Selecting an invalid entry shows a detailed parsing error on the right-hand panel.



```
Error at line: 173 column: 3 file: gfx\particles\_disr_projectiles_particles.gfx. Please report this error to
the author of the mod: Amazing Space Battles.

Error message says:

Error in gfx\particles\_disr_projectiles_particles.gfx: Ln: 173 Col: 10
(UTF16-Col: 3)
        }
         ^
Note:
* The column count assumes a tab stop distance of 8 chars.
* The error occurred at the end of the input stream.
Expecting: closing brace, statement or whitespace
Other error messages:
  The clause opened at ("gfx\\particles\\_disr_projectiles_particles.gfx", Ln:
  1, Col: 15) was not closed.
```

Invalid definitions often indicate:

- Mistyped syntax

- Broken scopes
- Missing brackets
- Misplaced keywords

These issues may prevent a mod from working correctly in-game.

5. Developer Overrides

Mod developers can instruct Irony how to treat certain files or definitions.

Irony supports three special comment directives:

1. Fallback to Simple Parser

```
### Dear Irony please fallback to simple parser
```

Forces Irony to bypass CWTools parsing for this file and use a less strict parser.

2. Ignore Entire File (placeholder)

```
### Irony this is a placeholder file please ignore it
```

Irony will ignore duplicate detection inside this file.

3. Ignore Specific Definitions (placeholder objects)

```
### Irony these are placeholder objects please ignore them: id1,id2
```

Only the listed definition IDs will be ignored.

Important Note

If Irony detects that your placeholder object would **win a conflict** in a FIOS/LIOS scenario,
it may override your ignore directive to avoid giving placeholders unintended priority.

This is intentional to prevent mod load order issues.

6. Tips for Modders

- Use placeholder ignore comments for prototype or empty files
- Use fallback parser for files with non-standard syntax
- Validate definitions inside CWTools errors to ensure compatibility
- Use the Conflict Filter to focus on important conflicts → Conflict Filter
- If you ran the Conflict Solver once and changed your collection, run it again to refresh patch mods → Conflict Solver Modes

Summary

The **Conflicted Objects** view provides:

- A hierarchical overview of all detected conflicts
- Direct access to definitions that require attention

- Visual reset indicators for changed items
- Tools to inspect invalid definitions
- Developer-side control via Irony directives

This section is the core of conflict analysis in Irony and the main entry point to the Merge Viewer and further conflict resolution tools.

Merge Viewer

The **Merge Viewer** is the central tool used to inspect, compare, and resolve conflicts detected by the Conflict Solver → Conflict Solver Modes.
It allows you to analyze differences between mods, understand which definition "wins," and build a merged version using a virtual definition.

1. Overview

The Merge Viewer opens when you select a conflicted definition inside **Conflicted Objects** → Conflicted Objects.
It presents:

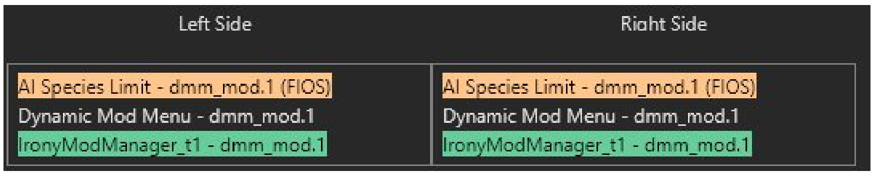
- All definitions contributing to the conflict
- A virtual, editable definition
- Comparison tools for navigating and understanding differences

The Merge Viewer may operate in **editable** or **read-only** mode depending on the capabilities of the underlying conflict data for that specific definition.

2. Definitions Pane

The left side lists all **definitions** contributing to the conflict.

Example screenshot:



Left Side	Right Side
AI Species Limit - dmm_mod.1 (FIOS)	AI Species Limit - dmm_mod.1 (FIOS)
Dynamic Mod Menu - dmm_mod.1	Dynamic Mod Menu - dmm_mod.1
IronyModManager_t1 - dmm_mod.1	IronyModManager_t1 - dmm_mod.1

Each definition is color-coded:

□ Green — Virtual Definition

This is the **editable** definition.
All merge actions apply to this virtual definition.

The virtual definition is created by copying the content of the **orange** definition, unless conflict history is available.

□ Orange — Chosen Game-Winning Definition

This is the definition that Irony believes the **game will actually use** if you do not resolve the conflict.

It may display labels such as:

- **FIOS**
- **LIOS**

- **Load Order**
- **Override**

These labels indicate **why** this definition is expected to win in-game under the current load order and dependency rules.

Irony uses this definition as the starting point for the virtual definition, unless conflict history is available.

3. Merge Viewer Actions (Non-Virtual Definitions)

Non-virtual definitions are **read-only**.
You cannot edit their contents directly.

Available actions:

• Next / Prev conflict

Moves to the next or previous difference.

• Copy text

Copies the entire definition to the clipboard.

• Copy all

Copies **all code blocks** from the selected definition into the virtual definition.

• Copy this

Copies only the selected code block into the virtual definition.

• Copy this before line

Inserts the selected block **above** the highlighted block in the virtual definition.

• Copy this after line

Inserts the selected block **below** the highlighted block in the virtual definition.

4. Merge Viewer Actions (Virtual Definition)

The virtual definition (green) is where all merging takes place.

Available actions:

• External Merge

If an external merge tool is configured in **Options** → Options,
Irony can launch it with the current conflict.
Irony waits for you to finish editing and **save** changes in the external tool before importing them back.

• Toggle Diff Compare Mode

Switches between the new diff interface and the legacy viewer.

• Next / Prev conflict

Navigates between conflict points inside the virtual definition.

• Copy text

Copies the virtual definition to the clipboard.

• Edit

Allows direct editing of the virtual definition.

• Move up / down

Moves the selected code block within the virtual definition.

5. Resolve & Ignore

Resolve

Writes the virtual definition to the **patch mod**.
The conflict disappears from the Conflicted Objects view → Conflicted Objects

A conflict reappears if:

1. The mod collection changes
2. A mod affecting the definition is updated
3. You manually **Reset Conflicts** → Reset Conflicts

Ignore

Marks the conflict as ignored.
It remains hidden until the underlying content changes again.

6. Back

Returns to the main Irony interface without resolving or ignoring the conflict.
Edits to the virtual definition are preserved until the solver is closed or the collection changes.

7. Read-Only Scenarios

Merge Viewer operates in **read-only mode** when:

- The definition type cannot be auto-merged
- The conflict is being analyzed only
- The data structure does not support merge operations

You may still:

- Navigate conflicts
- Copy text
- Inspect differences

…but you cannot modify or merge them.

Summary

The Merge Viewer provides:

- A detailed comparison of all definitions involved in a conflict
- A virtual definition for building a resolved version
- Tools for selective merging and navigation
- Optional integration with external merge tools

- Resolve/Ignore workflows
- Support for both editable and read-only scenarios

It is the primary interface for analyzing and resolving conflicts in Irony.

Binary Merge Viewer

The **Binary Merge Viewer** is used when Irony detects a conflict involving a **binary file**, such as a texture, model, sound, or any non-text game asset.
Binary files cannot be diffed or merged like script files, so Irony provides a minimal, focused interface for selecting which version of the binary file should be used in the final patch mod.

1. Overview

Binary Merge Viewer opens automatically whenever a conflicted file is **not text-based**.
Examples include:

- `.dds` , `.png` , `.tga` textures
- Audio files
- Models
- Any file format that cannot be parsed as text

Unlike the main Merge Viewer:

- **No virtual definition is created**
- **Binary files cannot be edited inside Irony**
- The viewer only allows choosing which mod’s binary file should be used

See the main Merge Viewer documentation for general behavior:
→ Merge Viewer

2. Definitions

The list of definitions behaves the same as in the main Merge Viewer, except for one important difference:

- **Binary conflicts do not have a virtual definition.**

Every listed entry corresponds to the binary file as provided by a specific mod.

See the regular Merge Viewer page for how definitions work:
→ Merge Viewer

3. Preview (Images Only)

As of Irony **1.6**, the Binary Merge Viewer attempts to show a **preview** for supported image formats.

- If the file is a texture (DDS, PNG, TGA…), Irony displays its preview
- If the image cannot be displayed, or the format is unsupported, a blank placeholder is shown

If an image should preview but doesn't, report:

- Mod name
- Definition name / file path

Other binary files (audio, models, etc.) **cannot** be previewed and are listed as binary blocks only.

4. Choosing a Binary File (Take Left / Take Right)

Binary conflicts are resolved by choosing **which version of the file should win**.

Available actions:

• Take Left

Selects the left-side binary file as the output.

• Take Right

Selects the right-side binary file as the output.

Only **one** binary file can be chosen.

Once you select Left or Right, the **Resolve** button becomes available.

5. Resolve

Clicking **Resolve** performs the following:

- The selected binary file is written directly into the **patch collection mod**
- The conflict disappears from the Conflicted Objects list → Conflicted Objects
- It will reappear only if:
  1. The collection changes
  2. A mod updates the binary file
  3. You reset conflicts manually → Reset Conflicts

Binary files cannot be merged—only selected.

6. Limitations

Binary Merge Viewer does **not** support:

- Editing
- Text diff
- Virtual definitions
- Block-based merge actions
- External merge tools
- Fine-grained conflict resolution

Binary conflict resolution is binary in the literal sense:
choose left or choose right.

Summary

Binary Merge Viewer provides:

- A simple interface for resolving conflicts involving binary assets
- Image previews for supported textures

- Quick Left/Right selection
- Patch-mod output identical to regular conflict resolution

This tool is essential for resolving asset overrides and ensuring the final game uses the correct binary resources.

Conflict Solver Modes

The Conflict Solver offers multiple modes that determine how Irony detects, displays, and processes mod conflicts.
Each mode changes how strictly conflicts are interpreted and whether Irony is allowed to generate patch output.

Modes fall into three families:

- **Default**
- **Advanced**
- **Analyze Only**

Each mode also has a **No Localization** variant.

1. Overview

When starting the Conflict Solver, Irony presents a mode selection dialog.
This determines:

- How conflicts are interpreted
- Whether dependency information is respected
- Whether patch output is generated
- Whether localization files are included
- How much detail is shown

Irony will prompt for mode selection every time you run the solver unless conflict history and state allow an automatic continuation.

2. Mode Selector Dialog

The dialog provides:

- A mode selector (Default / Advanced / Analyze Only)
- A toggle to exclude localisation folders ("Without Localization")
- A short description of each mode

After choosing a mode, Irony loads and analyzes all affected definitions.

3. Default Mode

Default Mode is designed for normal use and hides noise from expected overriding behavior.

Behavior:

- Respects `.mod` descriptor **dependencies**
- If Mod A supersedes Mod B via dependencies, Irony suppresses conflicts coming from B
- Shows only meaningful conflicts

- Reduces noise for typical playset management
- Produces patch output when conflicts are resolved

Use case:

Recommended for all non-modders and for regular modlist maintenance.

4. Default (No Localization)

Same behavior as Default Mode, but excludes localization files entirely.

Behavior:

- Skips all `localisation/` folders
- Useful when localization overrides are irrelevant or intentionally noisy

5. Advanced Mode

Advanced Mode bypasses dependency pruning entirely.

Behavior:

- **Ignores dependencies**
- Shows *all* conflicts, including:
    - Overrides normally hidden due to “A replaces B” relationships
    - Hidden conflicts masked by load order
- Provides maximum detail
- Generates patch output when conflicts are resolved
- Useful for mod developers

Use case:

Full visibility for debugging, creating compatibility patches, or validating large overhauls.

6. Advanced (No Localization)

Same as Advanced Mode, but excludes all localization files.

Behavior:

- Full conflict detail
- No localisation noise
- Faster analysis in localization-heavy modlists

7. Analyze Only

Analyze Only Mode loads conflicts but **never produces patch output**.

Behavior:

- Fully read-only
- All conflict information is visible

- Same UI as full modes (except editing is disabled)
- Useful for debugging or checking overrides without altering anything

Use case:

- Debugging large modsets
- Verifying mod interactions without creating a patch mod
- Cross-checking mod compatibility before making changes

8. Analyze Only (No Localization)

Same as Analyze Only, but excludes localization folders.

Behavior:

- Read-only
- No localisation noise
- Faster analysis for content-focused debugging

9. When Irony Prompts Again

Irony tracks the internal state of each conflict.
Even after selecting a mode, Irony will ask again under these conditions:

- Conflict state changed
- New conflicts were introduced
- Automatic reset happened
- The collection changed (mod added/removed)
- A mod updated a file (hash mismatch)

Irony **does not automatically invalidate or delete** patch output.
Instead:

✔ Irony marks the patch as “dirty”

- An updated mod or a collection change triggers a state mismatch
- The affected definition receives a **yellow or red border** depending on previous state
- Irony displays a notification such as “Please rerun Conflict Solver”

Patch output remains intact until the user resolves again or manually resets.

More info:
→ Reset Conflicts

10. Background Change Detection

Irony automatically scans mod structure in the background:

- File hashes
- Folder structure
- Definition hashes

- Contributor presence

If anything changes, Irony flags conflicts that may need attention.

Patch files are **not** removed automatically unless the conflict becomes irrelevant (no contributors), in which case Irony cleans old output silently.

11. When to Use Each Mode

Use Default Mode

- Normal gameplay
- Regular modlists
- Minimal noise
- Fastest workflow

Use Advanced Mode

- Debugging tricky mods
- Creating compatibility patches
- Investigating override chains

Use Analyze Only

- Verifying mod interactions
- Multiplayer consistency checks
- Validating modset without touching patch output

Use “No Localization”

- When localization overrides generate unnecessary noise
- When working only with gameplay files

12. Summary

Conflict Solver Modes define how Irony interprets and displays conflicts:

- **Default** respects dependencies
- **Advanced** shows everything
- **Analyze Only** is read-only
- “**No Localization**” variants exclude localization folders
- Irony warns about changes with visual indicators and prompts, but never removes patch data unless it becomes irrelevant

Choosing the right mode ensures the best balance between precision, clarity, and performance.

Ignore Rules

Ignore Rules allow you to hide specific conflicts from the Conflict Solver based on path patterns, mod names, or filtering conditions.
They are useful when you want Irony to skip certain folders, definitions, or entire mods that you know are safe to ignore.

1. Overview

Ignore Rules are different from simply clicking **Ignore** on a single conflict.

- **Ignore** (button) hides one specific conflict until the underlying files change.
- **Ignore Rules** define *persistent patterns* that automatically hide all matching conflicts every time the Conflict Solver runs.

Use Ignore Rules when you consistently want Irony to exclude certain files or mods from the conflict list.

Read more about the Conflict Solver here:
→ Conflicted Objects
→ Conflict Filter

2. Opening Ignore Rules

In the Conflict Solver window, the **Ignore Rules** button is located next to the **Ignore** button.

Clicking it opens a text editor where you can define rules using a simple syntax.

After saving, the Conflict Solver automatically refreshes with your updated rules applied.

3. Syntax

Ignore Rules support several syntax types.
Each rule must be on its own line. A line is either **a rule** or **a comment**, but never both.

3.1 Comments

Start a line with `#` to write a comment:

```
### This is a comment explaining why a rule was added.
```

Comments are ignored by Irony but useful for documenting your rules.

3.2 Basic Path Prefix Matching

A rule matches conflicts if the file path **starts with** the text you provide:

```
events
```

This hides all conflicts under the `events` folder.

3.3 Subfolder Rules (Backslashes)

You can specify full or partial folder paths:

```
common\defines
```

3.4 Linux-Style Paths (Forward Slashes)

Forward slashes are also supported:

```
common/defines
```

Both styles are equivalent.

3.5 Negation (Allowing Subfolders)

You can allow exceptions using the `!` prefix:

```
!common\defines\subfolder
```

This tells Irony to **include** that subfolder, even if the main rule hides everything else under `common\defines`.

3.6 Excluding Mods by Name

You can exclude all conflicts coming from a specific mod:

```
modname:Some Mod Name
```

This hides conflicts where the contributing definition originates from that mod.

3.7 Count-Based Exclusion (Advanced)

As of Irony **1.25**, you can combine mod exclusion rules with a count condition:

```
modname:Some Mod Name--count:3
```

This means:

- If the conflict involves **fewer** than 3 definitions, the mod is excluded.
- If the conflict involves **3 or more** definitions, it is not excluded.

This interacts with the Conflict Filter's "count settings."

→ Conflict Filter

3.8 Wildcards

Wildcards allow matching file name patterns:

```
localisation\*l_german.yml
```

This hides **all German localisation files**, regardless of subfolder depth.

4. How Irony Applies Ignore Rules

Irony processes rules as follows:

- Paths are matched using **starts-with** logic.
- Rules apply before conflicts are displayed.
- Exceptions (`!path`) override earlier path rules.
- Mod exclusions always apply (subject to count rules).

- Wildcards allow simple pattern matching for filenames.
- Ignore Rules work **alongside** the Conflict Filter.

If multiple rules overlap, the most specific rule applies.

5. Saving & Refreshing

Click **Save** in the Ignore Rules editor and Irony will immediately:

1. Re-run the conflict evaluation
2. Apply all matching rules
3. Refresh the Conflict Solver interface

No restart is required.

To remove a rule, simply delete its line and save again.

6. Sharing Ignore Rules

You can share your Ignore Rules configurations or see examples from other users in the community discussion:

https://github.com/bcssov/IronyModManager/discussions/156 (https://github.com/bcssov/IronyModManager/discussions/156)

7. Best Practices

- Use comments to document why a rule exists.
- Prefer folder-level ignores to long lists of specific files.
- Avoid ignoring definitions unless you're sure they are harmless.
- Use negation rules (`!`) to fine-tune large folder ignores.
- Use count-based rules for reducing noise in complex modlists.

Ignore Rules give you full control over how Irony filters conflicts, allowing a cleaner and more focused conflict-solving workflow.

Reset Conflicts

The **Reset Conflicts** tool allows you to manually reset the state of conflicts that were previously **resolved** or **ignored**, returning them to a fresh unresolved state.
This tool works alongside Irony's **automatic reset and cleanup system**, which is triggered when underlying mod data changes.

1. How to Use Reset Conflicts

1. Open the **Conflict Solver**
2. Click the **Reset Conflicts** button
3. Choose the conflict type:
    - **Resolved**
    - **Ignored**

4. Select the folder containing the conflict
5. Select one or more definitions
6. Click **Reset**

Multi-select is supported using Shift / Ctrl.

Related pages:
→ Conflicted Objects
→ Conflict Filter

2. What Manual Reset Does

Manual reset clears Irony's stored decision for that conflict.

Resetting a Resolved Conflict

- The previously generated resolved output is **removed** from the patch mod
- The “resolved” state marker is cleared
- The conflict appears again as **new/unresolved** the next time the solver loads (if it still exists)

Use this when you want to redo a merge manually.

Resetting an Ignored Conflict

- No patch files are affected
- The “ignored” marker is cleared
- The conflict appears again as **new/unresolved**

Use this when you want to review a previously ignored conflict.

3. Automatic Reset (Irony’s Smart System)

Irony automatically resets conflicts when it detects that the underlying data has changed.
This includes:

- A **mod update** changed a file
- The **definition hash** no longer matches the stored resolved state
- The **conflict history** no longer matches
- The **collection structure** changed (mod added/removed), affecting relevance
- A resolved output no longer has any contributing mod (no longer referenced)

In these cases:

- Irony automatically discards the old resolution
- The conflict is shown again so the user is aware something changed
- A border indicator appears (see below)

Automatic reset ensures that outdated resolves or ignores do not silently remain in the patch mod.

4. Automatic Cleanup (No Conflict Remaining)

If the underlying conflict disappears completely — for example:

- a mod is removed
- the file or definition no longer exists
- no mods contribute the conflicted object anymore

Irony will **automatically remove** the resolved output from the patch mod, because it is no longer relevant.

In this scenario:

- No conflict is shown
- No border indicator appears
- Patch output is silently cleaned up

This behavior prevents “dangling overrides” in the patch collection.

5. Visual Indicators (Automatic Reset Only)

In the **Conflicted Objects** listbox:

- **Red border** → automatically reset previously *resolved* conflict
- **Yellow border** → automatically reset previously *ignored* conflict

These indicators appear **only** when Irony performs an **automatic reset**.

Manual reset does **not** show any border.

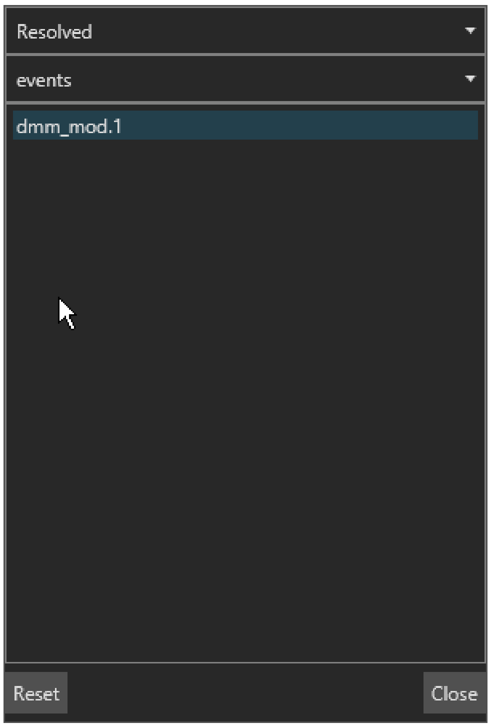
6. When to Use Manual Reset

Use manual Reset Conflicts when:

- You want to redo a merge by choice
- You want to clear a specific resolved entry from the patch mod
- You previously ignored a conflict and now want to handle it
- You want a clean manual starting point before merging
- You want to reset **only selected** conflicts (not all)

Do **not** use manual reset to handle mod updates — Irony already handles these automatically.

7. Screenshot



Reset Conflicts provides a precise, selective way to clear outdated decisions, while Irony's automatic system ensures merge consistency whenever mods or definitions change.

Database Search

The **Database Search** tool allows you to search through Irony's internal metadata database built during the mod scanning phase.
This system is extremely fast because Irony indexes **only IDs**, not full file contents.

Database Search is ideal for quickly finding which mods define or override specific scripted objects.

1. What Irony Indexes

When Irony parses your mod collection, it builds a metadata database containing:

- **Definition IDs**
  Examples:
  - `anomaly.502`
  - `anomaly.555`
  - `my_scripted_effect`
  - `start_screen_event.20`
- **Definition names** (if available)
- **File paths** where the definitions were found
- **Mod source** for each definition

This information is collected from all supported script formats Irony can parse.

□ Important

Irony does **not** index full code bodies or arbitrary text.
It indexes only **IDs** extracted from parsed definitions.

2. What Database Search Is Used For

Database Search is perfect for answering questions like:

- “Which mod defines event `anomaly.555` ?”
- “Which mods contain a scripted effect named `my_effect` ?”
- “Are multiple mods overriding the same ID?”
- “Where is this definition located across the entire load order?”
- “Which mod introduces this specific object into the game?”

This is extremely useful when analyzing conflicts, creating compatibility patches, or checking if a mod unexpectedly overwrites something.

3. How to Use Database Search

1. Open the **Database Search** tool
2. Type the ID or partial ID you want to find
3. Irony displays a list of all matching items

The search is nearly instant, regardless of how many mods are installed.

Supported search patterns:

- **Exact ID**
- **Partial ID** (prefix or substring)
- **Case-insensitive** matching

4. Search Results

Each result typically displays:

- **ID**
- **File path**
- **Mod name**
- **Source type** (local, workshop, paradox)

This allows you to quickly identify:

- Where a definition comes from
- Which mods override the same object
- Whether you have duplicates
- Which mod “wins” based on load order

5. How It Differs From Conflict Solver

Database Search:

- Shows **all IDs**, even if no conflict exists

- Is not tied to patch generation
- Is not affected by dependencies or conflict modes
- Does not show conflicts — only presence and location

Conflict Solver:

- Shows only **true definition-level conflicts**
- Is affected by modes, dependencies, and ignores
- Is used for merging and patch generation

Together, they form a complete analysis toolkit.

6. Use Cases

Database Search is especially useful for:

- Debugging unexpected gameplay behavior
- Finding duplicate IDs across mods
- Detecting scripted effects or events reused in multiple mods
- Checking if a mod actually contains what you think it contains
- Locating definitions for manual custom patching
- Validating mod structure when troubleshooting

Example:
Searching for:

```
anomaly.555
```

…shows *every mod* that defines or overrides that event.

7. Limitations

- Only IDs are indexed
- Full-text code search is **not** supported
- Malformed or invalid files may not provide extractable IDs

Summary

Database Search is a powerful, fast, ID-based index of your entire mod collection.
It allows you to instantly find which mods define or override specific objects without manually inspecting files.

It is essential for diagnosing overrides, locating definitions, and understanding mod interactions.

Conflict Filter

The **Conflict Filter** is a feature inside the Conflict Solver.
It controls which mods and which conflict types are shown, helping you focus only on what matters.

You can access it by toggling the **Mod Filter** button inside the Conflict Solver:
→ Conflicted Objects

What Mod Filter Does

Mod Filter allows you to temporarily hide conflicts coming from specific mods, based on the filter settings.

Example:
If a conflict involves **2 mods**, and your filter is configured to only show conflicts involving **3 or more mods**, then Irony will **not display** that conflict.

This is useful when reducing noise, especially in large mod lists.

Mod Filter Options

These options allow you to refine which conflicts are visible.

1. Ignore Game Conflicts (ON by default)

If enabled (default), Irony hides conflicts coming from **base game files**.

If disabled, Irony will show conflicts involving game definitions.
This significantly increases the number of visible conflicts.

Useful for:

- Updating an outdated mod to a newer game version
- Auditing changes between major patches (e.g., migrating a Stellaris mod from 3.2 → 3.3)

2. Show Self Conflicts (OFF by default)

Shows conflicts **within the same mod**.

Some mods declare the same object multiple times (intentional or accidental).
Turning this on helps detect those internal conflicts.

3. Show Only Reset Conflicts (OFF by default)

Shows only conflicts that:

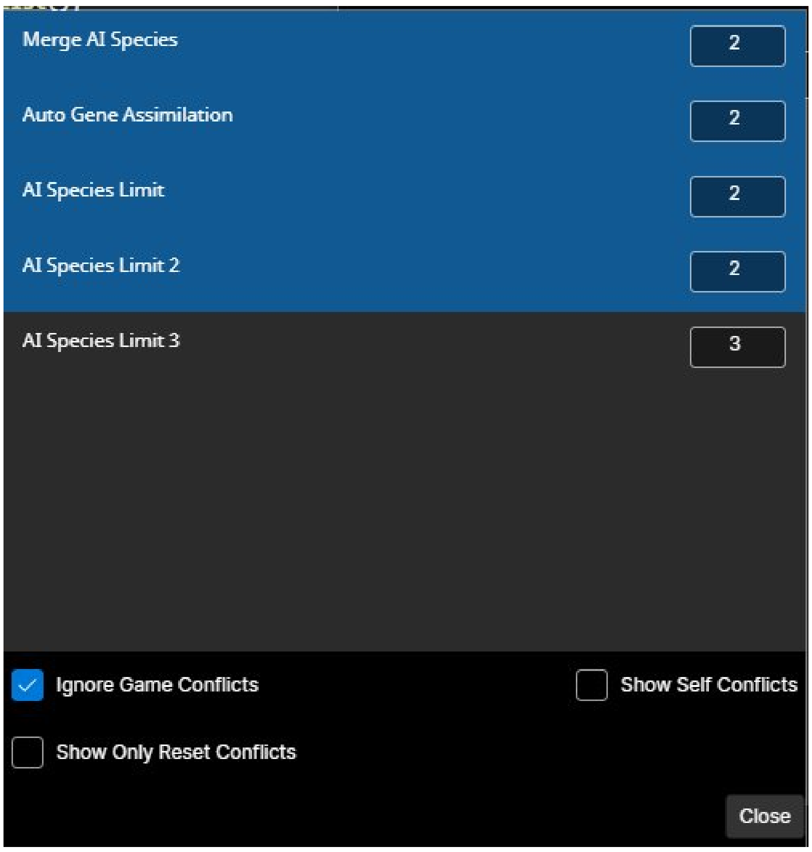
- **were reset**, or
- **will be reset**

(depending on the Conflict Solver mode)

All other Mod Filter settings are ignored when this option is active.

If nothing was reset, this option behaves as if it is turned off.

Screenshot


Merge AI Species 2
Auto Gene Assimilation 2
AI Species Limit 2
AI Species Limit 2 2
AI Species Limit 3 3

Summary

Conflict Filter helps you:

- Reduce noise by hiding irrelevant conflicts
- Show only conflicts that match the chosen criteria
- Include or exclude base game conflicts
- Detect self-conflicts inside a single mod
- Focus only on resettable conflicts when needed

It is a valuable tool for mod maintainers and advanced users working on large load orders or resolving complicated conflicts.

Custom Patches

Custom Patches allow you to add your **own manual fixes** directly into the Irony Patch Mod.
They are intended for small, targeted corrections that you want Irony to respect during conflict analysis without creating a separate standalone mod.

This feature is ideal when you want to override or repair a definition quickly and locally.

See also:
→ Conflict Solver
→ Invalid Objects

1. What Custom Patches Are For

Custom Patches let you:

- Inject your **own code overrides** into the Irony Patch Mod

- Apply quick fixes without building a separate mod
- Override definitions that Irony would not normally merge
- Fix **invalid definitions** detected by CWTools
- Adjust or correct problematic lines in a mod author's file
- Make temporary or experimental adjustments

They give you full control when you need a small fix *now*, without waiting for a mod update.

2. Recommended Use Cases

Use Custom Patches when:

- A mod contains a broken or malformed definition
- Irony marks something as **Invalid** but you know how to fix it
- You want to override a small piece of data for your own playthrough
- The change is too minor to justify creating a separate mod
- You want Irony to treat your custom override as part of the analysis

For larger edits or long-term maintenance, it is recommended to create a dedicated external mod instead.

3. Custom Patches and Invalid Definitions

Custom Patches fully support overriding **Invalid** definitions.

Steps:

1. Locate the invalid definition in the "mod folder" selector
2. Write your corrected version in the Custom Patch editor
3. Save the patch
4. Rerun the Conflict Solver

Irony will then:

- Prefer your custom override over the broken definition
- Treat your patched version as the active definition during conflict analysis

4. How Custom Patches Integrate With Irony Patch Mod

When you save a Custom Patch:

- Irony writes your override into the Irony Patch Mod
- The custom file is treated like any other resolved definition
- During analysis, Irony uses **your version** instead of the original
- The patch persists until manually edited or removed

Custom patches **do not** get auto-resolved or auto-reset like normal conflicts — they exist only because *you* created them.

5. Important Notes

- After creating or editing a Custom Patch, you **must rerun the Conflict Solver**
- Custom Patches override the original mod's data during analysis and merging
- They are intended for **small-scale adjustments**, not full modding
- If the underlying mod updates, your patch may need revisiting
- Custom Patches are stored inside Irony's patch mod, not as a separate mod

Summary

Custom Patches provide a lightweight way to fix or override definitions directly inside the Irony Patch Mod.
They are perfect for quick corrections, invalid definition fixes, and small personal tweaks — without needing to build a standalone mod.

Use them for precision fixes, and use external mods for large-scale changes.

Options

The Options panel contains all global and game-specific settings for Irony Mod Manager.
These settings control language, themes, game paths, editor integration, update behavior, and advanced conflict solver options.

1. General Options

Language

Selects the application UI language.

Themes

Choose between available visual themes for the Irony UI.

Game

Changes the currently active game.
All game-specific settings below apply only to the selected game.

Actions

Provides quick access to useful tools:

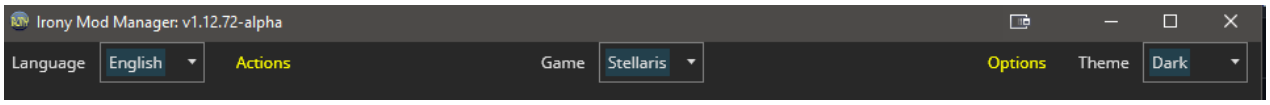
- **Open DLC Manager**
- **Open Irony Wiki**
- **Open Irony Logs Directory**
- **Open Selected Game's `error.log`**

2. DLC Manager

Allows enabling or disabling DLCs for the selected game.

After adjusting DLC checkboxes, press **Close** to apply.

3. Options Screen Overview



The options screen is divided into several categories:

- Game Options
- External Editor Options
- Conflict Solver Options
- Mod Merge Options
- Update Options

Each section is described below.

4. Game Options

Configure All

Automatically detects game paths and settings when you select the **root installation folder**.

Irony attempts to determine:

1. Game executable
2. Game data directory (e.g. `Documents/Paradox Interactive/Stellaris`)
3. Launch arguments

Note: Irony does not require selecting the exact root folder.
If you select a deeper folder (e.g. `.../Stellaris/common`), Irony walks upward until it finds a valid game structure.

Game Executable

Path to the file Irony should launch when “Launch Game” is pressed.

Game Args

Additional command-line arguments to pass to the game executable.

Data Directory

Location where the game stores:

- settings
- save files
- mods
- launcher data

Example (Stellaris):
`Documents/Paradox Interactive/Stellaris`

Custom Mod Directory

Adds an **extra** folder from which Irony will load mods.
If enabled, this folder also becomes the location for:

- merged mods

- patch mods

Refresh Descriptors Before Playing

Before launching the game, Irony:

- deletes existing mod descriptors
- regenerates them cleanly

Useful for stale or partially written `.mod` files.

Close After Launching Game

If enabled, Irony closes automatically after launching the game.
If disabled, it remains open.

5. External Editor Options

Used in the Conflict Solver to open the virtual definition in a third-party merge/diff tool.

Editor Executable

Path to the external editor binary (e.g. `winmerge.exe` , `code.exe`).

Editor Args

Arguments passed when launching the external editor.

Working examples may be shared here:
https://github.com/bcssov/IronyModManager/discussions/176 (https://github.com/bcssov/IronyModManager/discussions/176)

6. Conflict Solver Options

Allowed Languages

Enable/disable parsing of specific localization languages.

Disabling unused languages (e.g. German, Spanish) improves performance.

New Diff Viewer Colors

Customize colors used in the new diff viewer.

(Does not affect the legacy merge viewer.)

7. Mod Merge Options

Mod Template

Naming format for individually compressed-merge mods.
Must include:

- `{Name}`
- `{Merged}`

Collection Template

Naming format for merged collections.
Must include:

- `{Name}`
- `{Merged}`

8. Update Options

Check for Updates at Startup

Automatically checks Irony's update feed each time the application starts.

Update to Prerelease Versions

If enabled, Irony may update to:

- alpha
- beta
- rc
- stable

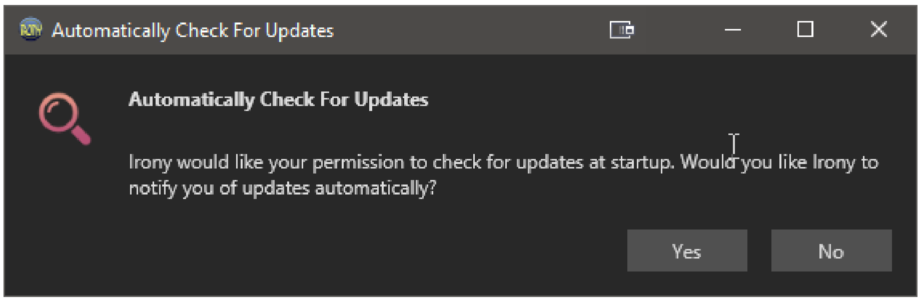
If disabled, only stable releases are used.

Download and Install

Begins the update process when a new version is available.

9. Understanding the Update Process

On first run, Irony displays:


Automatically Check For Updates

Automatically Check For Updates

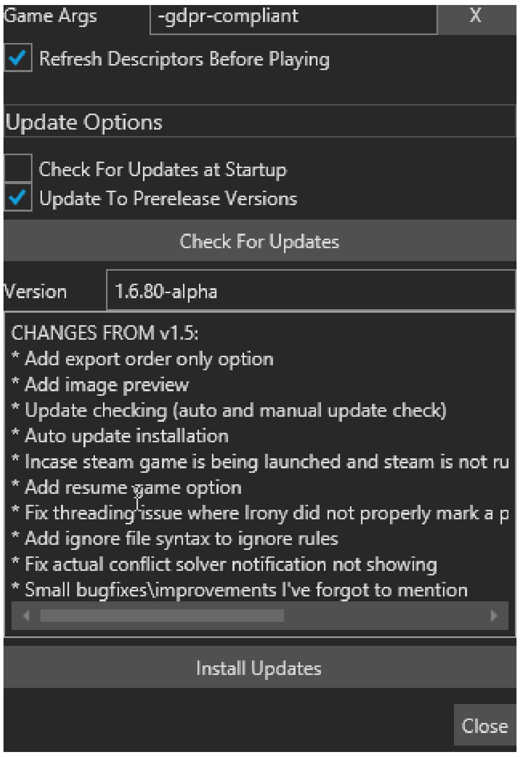
Irony would like your permission to check for updates at startup. Would you like Irony to notify you of updates automatically?

If allowed, Irony checks the feed:

```
https://bcssov.github.io/IronyModManager/appcast.xml
```

Which lists available releases.

When updates are found:


Game Args -gdpr-compliant X

✓ Refresh Descriptors Before Playing

Update Options

☐ Check For Updates at Startup
✓ Update To Prerelease Versions

Version 1.6.80-alpha

```
CHANGES FROM v1.5:
* Add export order only option
* Add image preview
* Update checking (auto and manual update check)
* Auto update installation
* Incase steam game is being launched and steam is not ru
* Add resume game option
* Fix threading issue where Irony did not properly mark a p
* Add ignore file syntax to ignore rules
* Fix actual conflict solver notification not showing
* Small bugfixes\improvements I've forgot to mention
```

Download Behavior

Update packages are stored in:

```
%AppData%/Mario/IronyModManager-Updates
```

Files are signature-verified and then installed depending on installation type.

10. Installation Types

1. Portable Installation

- Zip is unpacked
- Console updater is launched
- Irony closes
- Updater copies new files into the install directory
- Irony restarts
- On macOS/Linux: executable permissions restored (`chmod +x`)

2. Installer Installation

- Zip is unpacked
- Setup installer is launched
- Update applies via installer UI

Update files remain for ~3 days before automatic cleanup.

Summary

The Options screen centralizes all configuration for paths, editor tools, updates, conflict solver behavior, and merge templates.

Most users configure it once; advanced users may fine-tune tools and parsing behavior to suit their workflow.

Keyboard Shortcuts

Irony provides a wide range of keyboard shortcuts to speed up navigation, conflict resolution, and mod ordering.
Shortcuts react to the **currently active window** and often to the **hovered item**.

1. Main Screen

Collection Mods Area

Shortcuts apply to the mod currently **hovered** or selected in the Collection Mods list.

Hotkey	Action
`CTRL + UP`	Move mod up
`CTRL + DOWN`	Move mod down
`CTRL + SHIFT + UP`	Move mod to top
`CTRL + SHIFT + DOWN`	Move mod to bottom
`CTRL + Z`	Undo last change
`CTRL + Y`	Redo undone change

* Movement operations update the collection order immediately.

2. Conflict Solver

These shortcuts apply inside the Conflict Solver, including the Merge Viewer.

Navigation

Hotkey	Action
`CTRL + UP`	Jump to previous conflict
`CTRL + DOWN`	Jump to next conflict
`CTRL + LEFT`	Previous conflict (skip empty/imaginary lines)
`CTRL + RIGHT`	Next conflict (skip empty/imaginary lines)
`SHIFT + UP`	Select next conflict in the tree
`SHIFT + DOWN`	Select previous conflict in the tree
`CTRL + SHIFT + UP`	Scroll Merge Viewer up
`CTRL + SHIFT + DOWN`	Scroll Merge Viewer down

Definition Selection (Merge Viewer)

Hotkey	Action
`CTRL + 1-0`	Select definition for the **left** side (1 = first definition)
`CTRL + SHIFT + 1-0`	Select definition for the **right** side (1 = first definition)

Conflict Actions

Hotkey	Action
`CTRL + I`	Ignore conflict
`CTRL + R`	Resolve conflict
`CTRL + E`	Enter edit mode for the virtual definition
`CTRL + X`	Launch external editor (if configured)

Text / Block Copy Actions

Hotkey	Action
`CTRL + T`	Copy text from the **left** side
`CTRL + SHIFT + T`	Copy text from the **right** side
`CTRL + C`	Copy selected block (“Copy This”)
`CTRL + V`	Insert block **before** selected line
`CTRL + B`	Insert block **after** selected line

Conflict Solver Undo / Redo

Hotkey	Action
`CTRL + Z`	Undo
`CTRL + Y`	Redo

Summary

Irony’s keyboard shortcuts are designed to streamline mod ordering, conflict navigation, and merge editing.
Learning just a handful of these dramatically speeds up workflow — especially when resolving a large number of conflicts.

Add Irony To Steam

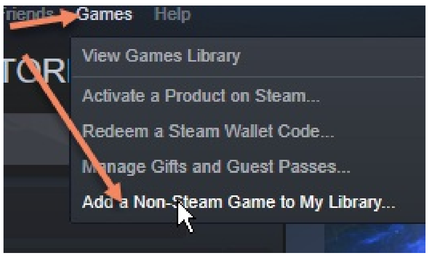
You can add Irony Mod Manager to Steam as a Non-Steam Game.
This is useful if you want:

- Quick access from your Steam library
- Big Picture Mode support
- Steam Deck integration
- Steam overlay behavior (Windows)
- Automatic game pre-selection when launching Irony

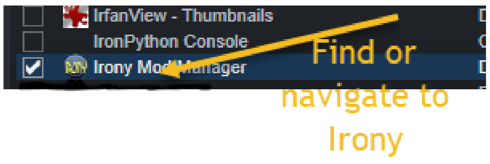
Irony works as a **native app** on Windows, Linux, and macOS.
On Linux, **Proton is NOT required** because Irony provides a native Linux binary.

1. Add Irony as a Non-Steam Game

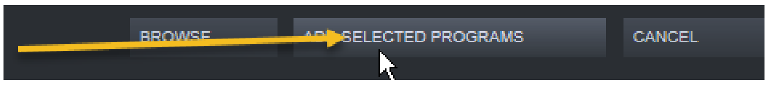
1. Open **Steam**
2. Go to **Games** → **Add a Non-Steam Game to My Library...**


Games
Help
View Games Library
Activate a Product on Steam...
Redeem a Steam Wallet Code...
Manage Gifts and Guest Passes...
Add a Non-Steam Game to My Library...

3. Select **Irony Mod Manager** from the list,
   or click **Browse...** and locate the executable:
   - **Windows:** `IronyModManager.exe`
   - **Linux/macOS:** `IronyModManager`



4. Click **Add Selected Programs**

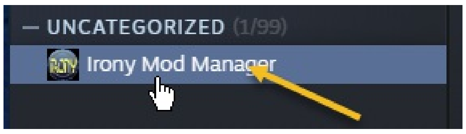


Irony will now appear in your Steam library.

2. Set Irony to Launch With a Specific Game Selected

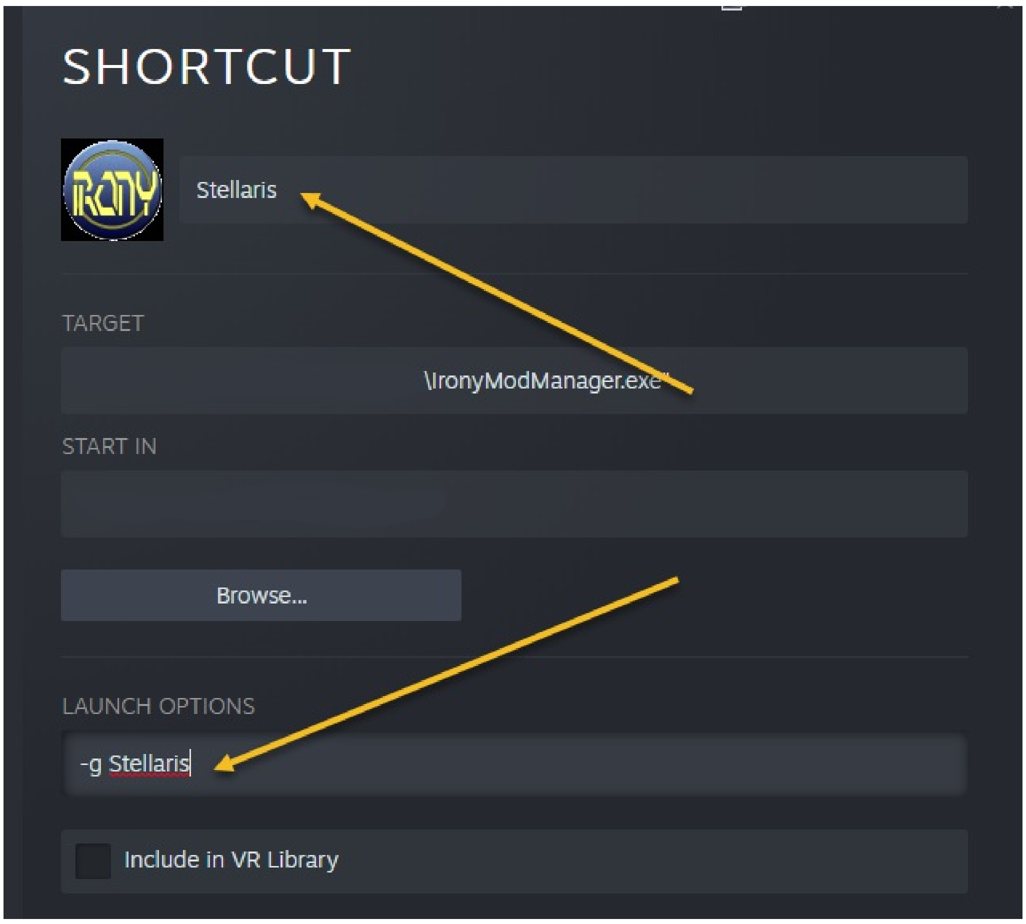
Steam allows you to customize the launch command so Irony opens *directly on the chosen game tab*.

1. In your Steam Library, right-click **Irony Mod Manager**
2. Select **Properties...**



3. Rename it to the game you want (optional).
4. In **Launch Options**, enter:

```
-g Stellaris
```


SHORTCUT

Stellaris

TARGET

\IronyModManager.exe"

START IN

LAUNCH OPTIONS

-g Stellaris

This tells Irony to start with Stellaris pre-selected.

3. Supported Game Launch Flags

Replace `Stellaris` with one of these identifiers:

- `CK3`
- `EU4`
- `HOI4`
- `IR`
- `Stellaris`
- `STInfinite`
- `Vic3`
- `EU5`

Example for Hearts of Iron IV:

```
-g HOI4
```

Example for Crusader Kings III:

```
-g CK3
```

4. Linux / Steam Deck Notes

- Irony has a **native Linux executable**, so **do not enable Proton**.
- Add Irony exactly like any other Linux-native program.
- Steam Deck can run Irony in Desktop Mode without compatibility tools.

Summary

Adding Irony to Steam improves workflow and lets you launch Irony directly into the game you're modding.
Just add it as a Non-Steam Game and optionally set a `-g <game>` launch flag for automatic game selection.

FAQ

This page answers common questions about load order, conflict solver behavior, mod metadata, Linux/macOS issues, updates, and troubleshooting.

1. Load Order & Mod Manager Behavior

Q: What load order does Irony use?

Irony uses the same load order as the **Paradox Launcher v2**.
Changes you make in Irony are applied directly to the game's mod registry in a launcher-compatible format.

Q: Does drag-and-drop support multiple items?

Not yet. Multi-select drag-and-drop is planned.
You can track the feature request here:
https://github.com/bcssov/IronyModManager/issues/12 (https://github.com/bcssov/IronyModManager/issues/12)

Q: I modified the collection — how do I clean up the patch mod?

You don't need to do anything manually.

- If you rerun the **Conflict Solver**, Irony will automatically clean outdated patch entries.
- You can also temporarily disable or delete the patch mod inside the **Collection Mods** panel.

2. Conflict Solver & Merging

Q: The conflict solver is missing for my game — why?

Only **Stellaris** has full conflict solver support.

- **HOI4** supports **Analysis Only** mode
- All other supported games use merging only

See the **Supported Games** page for details.

Q: Editing some conflicts is difficult. Any tips?

Some definitions are extremely complex.
You can configure an external merge tool in **Options** → **External Editor** (e.g., WinMerge, KDiff, VS Code) and use:

Right-click → **External Merge**

The external tool will open the virtual definition and allow full editing.

3. Mod Metadata & Invalid Definitions

Q: Irony reports "invalid definitions". Is this a problem?

Invalid definitions are detected using CWTools.
They usually indicate malformed syntax or unsupported structures in a mod.

You can:

- Inspect errors in the **Invalid** folder inside Conflict Solver
- Provide custom overrides via **Custom Patches**
- Fix your own mods directly

If you believe Irony incorrectly flagged something, you can report it.

Q: My mod is flagged incorrectly. Can I tell Irony to ignore a file?

Yes. Inside the file, add one of the following comments on a line by itself:

```
### Dear Irony please fallback to simple parser
```

or

```
### Irony this is a placeholder file please ignore it
```

or specify placeholder IDs:

```
### Irony these are placeholder objects please ignore them: id1,id2
```

These prevent the definition from being used in conflict evaluation.

4. Linux, macOS & Technical Environment

Q: Irony does not detect Stellaris (or another game) on Linux.

Some distros do not set `XDG_DATA_HOME` .
Irony depends on this path to locate Paradox game directories.

Setting this variable or launching the game once usually resolves the problem.

Q: Irony freezes or shows black windows on Linux.

Possible causes:

- Avalonia UI bugs on certain distros
- Wayland incompatibility
- Tooltips causing UI deadlocks

You can fix this by editing `appSettings.json` :

```
"Tooltips": { "Disable": true }
```

Or enable Wayland support:

```
"LinuxOptions": { "DisplayServer": "wayland" }
```

Ensure `xwayland` is installed if using X11 fallback.

Q: Irony fails to launch on macOS Catalina.

See:
https://github.com/bcssov/IronyModManager/issues/119 (https://github.com/bcssov/IronyModManager/issues/119)

This is related to Apple's notarization and quarantine system.

Q: I run out of RAM when exporting/merging on macOS.

macOS has a very low `ulimit` (256), limiting how many file handles Irony can use.

Workaround:

1. Copy `appSettings.json → appSettings.override.json`
2. Set `"UseFileStreams": true` under `"OSXOptions"`
3. Open Terminal → run:
   ```
   ulimit -n 200000
   ```
4. Launch Irony via Terminal:
   ```
   ./IronyModManager
   ```

You must repeat step 3 for each Terminal session unless you make the limit permanent.

5. Updates & Crashes

Q: Irony crashes unexpectedly or auto-update fails.

Most commonly caused by:

- Antivirus blocking Irony
- Incomplete downloads
- System-level restrictions

Digitally signed binaries would prevent false positives, but code signing is expensive.

Q: Irony is crashing on startup on Windows.

Install the Microsoft Visual C++ 2017 Redistributable:

- x86: https://aka.ms/vs/16/release/vc_redist.x86.exe (https://aka.ms/vs/16/release/vc_redist.x86.exe)
- x64: https://aka.ms/vs/16/release/vc_redist.x64.exe (https://aka.ms/vs/16/release/vc_redist.x64.exe)

6. Workshop & Distribution Behavior

Q: Can Irony upload mods to the Steam Workshop?

No. Irony is a mod manager and conflict solver only.
It cannot upload or publish Workshop mods.

Q: How do I "freeze" my game state?

Options:

1. **Merge** → **Compress**
2. **Merge** → **Basic**
3. Duplicate a collection → **Export** → **Whole Collection** → re-import later

This ensures all mods exist locally in their exact state.

7. Wayland / Linux Issues

Q: Irony shows only a black window under Wayland.

Fix options:

1. Install **xwayland**
2. Change `"DisplayServer"` to `"wayland"` or `"auto"` in `appSettings.json`

Q: Is there a Wayland-ready package in my distro's repository?

Some distros (e.g., Arch) offer community packages like `irony-mod-manager-bin` ,
but they are **not officially maintained**.
Report issues to the package maintainers, not Irony.

Summary

The FAQ covers common questions about load order, conflicts, metadata, Linux/macOS issues, updates, and mod management behavior.
If something is not covered here, check GitHub Discussions or open a new issue.

Privacy Policy

This document explains what data Irony Mod Manager does—and does not—collect.
Irony is designed with privacy in mind and does not transmit any personal data.

1. Software Privacy

Irony Mod Manager does **not** collect, store, or transmit:

- personal information
- hardware identifiers
- usage analytics
- system information
- mod list data
- configuration files

Irony only makes network requests **when checking for updates**, and these requests do not send any personal data.

What is transmitted?

The only information visible to remote servers is your **IP address**, which is standard for any HTTP request and handled directly by GitHub's infrastructure.
None of this information is collected by the Irony developer.

Update-related requests

Irony performs only two types of online requests:

1. **Update check** (`GET` request):

```
https://bcssov.github.io/IronyModManager/appcast.xml
```

2. **Version download (if you choose to update)**:

```
https://github.com/bcssov/IronyModManager/releases/download/v{version}/{filename}
```

These endpoints are hosted by GitHub.
No additional tracking or telemetry is performed.

Turning off update checks

Automatic update checks can be disabled at any time in:

Options → Update Options → Check for updates at startup

Manual update checks are user-triggered and optional.

2. Website Privacy

The Irony website and documentation pages (GitHub Pages, GitHub Wiki, GitHub Releases) do not collect any information on behalf of the Irony developer.

Any data gathered (such as IP logs or analytics) is collected **solely by GitHub**, according to GitHub's own privacy policies.

The Irony developer does not receive, access, or process any of this information.

3. Removing Irony & Clearing Local Data

If you wish to uninstall Irony:

Step 1 — Determine your installation type

Portable Version

Simply delete the folder where you extracted Irony.

Installer Version (Windows)

Open **Add/Remove Programs**, locate:

```
Irony Mod Manager v{version}
```

and choose **Uninstall**.

Step 2 — Remove local settings & update cache (optional)

Irony stores settings and downloaded updates in:

```
%AppData%\Mario
```

Deleting that folder will remove:

- user settings
- UI preferences
- update cache files

This step is optional but recommended for a clean removal.

Step 3 — Remove Irony patch mods (optional)

If you want to remove generated patch mods:

1. Open your game's mod directory
2. Delete any folders and `.mod` descriptors prefixed with:

```
IronyModManager
```

This will remove Irony-created patch collections without affecting other mods.

Summary

Irony Mod Manager does not collect or transmit personal data.
The only network actions performed are update checks and downloads from GitHub, both of which expose only standard HTTP request metadata (e.g., IP address) to GitHub—not to the developer.

Removing Irony is straightforward, and all user data can be deleted locally.