Decision modding - Hearts of Iron 4 Wiki

This is a community maintained wiki. If you spot a mistake then you are welcome to fix it.

Decisions and missions represent actions that the player can or must take, each one stored within a category. Decisions themselves are defined within /Hearts of Iron IV/common/decisions/*.txt files, while categories are defined within /Hearts of Iron IV/common/decisions/categories/*.txt. While it's typical to put the country's tag within the filename, it is actually irrelevant: files can take on any filename and there will be no difference in how they will be read.

Basic creation[编辑 | 编辑源代码]

Each decision must be stored within a category. Decision categories are defined within any /Hearts of Iron IV/common/decisions/categories/*.txt file. An incredibly basic decision category definition looks like the following:

```
my_decision_category = {
}
```

This will create the `my_decision_category` category, which can be used within decisions.

Afterwards, decisions can be created for that category in any /Hearts of Iron IV/common/decisions/*.txt file. Decisions are assigned to a category by putting them within that category's block, encompassed with the curly brackets, as in the following example:

```
my_decision_category = {
  my_decision_1 = {
  }
  my_decision_2 = {
  }
}
```

This will create the `my_decision_1` and `my_decision_2` decisions, with the default icon and no effect, and assign them to the `my_decision_category` category.

A decision's name according to the currently-turned on language can be defined in any localisation file, taking the decision's name as key, and the decision's name with _desc added in the end as the name for the localisation key for the description. As an example, the localisation for these decisions for the English language will look like the following within any /Hearts of Iron IV/localisation /english/*_l_english.yml file:

```
 my_decision_category:0 "My decision category"
 my_decision_category_desc:0 "My decision category's description"
 my_decision_1:0 "My decision"
 my_decision_1_desc:0 "My decision's description"
 my_decision_2:0 "My decision without a description"
```

Arguments for decisions and categories[编辑 | 编辑源代码]

The following arguments can be used in either decisions or decision categories, usually with similar effect.

Triggers[编辑 | 编辑源代码]

Usually, it is desirable to restrict the decision so that it might not always be possible to take, such as restricting it to a certain country. In order to do that, there are several trigger blocks, serving different purposes. A decision will only be available or visible if the corresponding conditions are met in both the category and the decision.

`allowed` is a trigger block that checks only at the game's start or when loading a save, primarily used to restrict a decision to a country (As `tag = BHR` or `original_tag = POL`) and/or a DLC (As `has_dlc = "One Step Back"`). If a decision's or a category's allowed is unfulfilled, it will never appear unless it becomes true on the save being reloaded or decisions themselves being reloaded. If left out, assumes to be always allowed. **This only checks once!**

```
allowed = {
  original_tag = BHR
}
```

`visible` is a trigger block that continuously checks every frame if allowed was met, required to make the decision or the entire category be visible in the decision selection screen. In case for targeted decisions, both FROM and ROOT can be checked here, but using target_trigger is recommended when possible for optimisation purposes. It is preferable to put country or DLC checks into allowed instead. **Does nothing for missions!**

```
visible = {
  is_subject = no
}
```

`available` is a trigger block that continuously checks every frame if visible was met, required to be possible to actually take the decisions. If false, the decision (or, in case of categories, decisions within) will remain visible (unless set otherwise), but will be greyed out and be impossible to complete. This is applied on top of the political power cost. Applied differently for missions.

```
available = {
  has_war = yes
}
```

Icon[编辑 | 编辑源代码]

The icon is the small picture displayed to the left of decision's or category's name. Icons use sprites, defined in any /Hearts of Iron IV/interface/*.gfx file, by default in `decisions.gfx`. Within a decision or a category, it is set with `icon = icon_name`.
Note that the game automatically inserts either `GFX_decision_` or a `GFX_decision_category_`, depending on whether it's in a decision or a category. As such, the example with icon_name will make it use the `GFX_decision_icon_name` sprite for decisions and the `GFX_decision_category_icon_name` for categories, or, other way around, to use `GFX_decision_sprite_name` in a decision, it should have `icon = sprite_name`.
However, `icon = GFX_decision_icon_name` will also work, as the game will check the spriteType with the exact same name as well. Overall, it's to the developer's discretion whether to include `GFX_decision_/GFX_decision_category_` in the icon or not.

Priority[编辑 | 编辑源代码]

Priority is used to change the order in which decisions or categories are displayed from top to bottom, with a higher priority being closer to the top. By default, a decision has the priority of 1.
Decision priority can be set in a short form for a static value or in a long form, similar to ai_will_do formatting.
In short form, the following code will set a decision or category to have the priority value of 10: `priority = 10`
In long form, the following code will have a priority value of 13 for POL and 3 for other countries:

```
priority = {
```

```
    base = 3
    modifier = {
        add = 10
        tag = POL
    }
}
```

State highlighting[编辑 | 编辑源代码]

A decision or all decisions within a category can be set to highlight a state or several with an outline when hovering over it in the selection menu. This is the code required to do that:

```
highlight_states = {
  highlight_state_targets = {
    state = 123
    state = 321
  }
  highlight_color_while_active = 3
  highlight_color_before_active = 2
}
```

`highlight_state_targets` selects a specific state or several that will be highlighted. If the state is unknown, then `highlight_states_trigger` can be used as a trigger block instead, evaluated for every state on the map.

`highlight_color_before_active` is the colour that the state highlighting will have before selecting the decision, from 0 to 3. If not set, it will default to a white outline.

`highlight_color_while_active` is the colour that the state highlighting will have after selecting the decision during the timer before it gets removed, from 0 to 3. If not set, it will default to a white outline.

Within regular decisions, only one state can be highlighted at a time. This restriction does not exist when used inside of decision categories.

Arguments for categories[编辑 | 编辑源代码]

These can only be used within categories and cannot be used in separate decisions.

Picture[编辑 | 编辑源代码]

A decision category can have a picture defined in addition to its regular icon. A picture will show up to the left of the category's description, shifting it to the left. Pictures use sprites, defined in any /Hearts of Iron IV/interface/*.gfx file, by default in `decisions.gfx`. A sprite with the name of `GFX_decision_category_picture` can be set as the category's picture with `picture = GFX_decision_category_picture`. **A category must have a description defined in localisation for the picture to appear.**

Visibility when empty[编辑 | 编辑源代码]

By default, a decision category is invisible unless there's at least one visible decision within. This isn't always helpful, as the category may display information vital for the player in its description. To override that, you can add a line consisting of `visible_when_empty = yes` within the decision category.

Map area[编辑 | 编辑源代码]

A decision category can be assigned a map area, which will show up in the top of the decision list similarly to a decision. Clicking on the button will move the camera to the specified state or states, while setting the zoom level to the set amount. This is recommended to do in decision categories containing decisions targeted towards states. A map area definition looks like the following:

```
on_map_area = {
  state = 123
  name = my_localisation_key
  zoom = 850
  target_root_trigger = {
    tag = ENG
  }
}
```

`state` sets a singular state that's used as the centre of the area where the camera will move. If the map area is to be dynamic, then formatting similar to targeted decisions can be used, with `targets`, `target_array`, and `target_trigger` being available and mixable.
`name` is the localisation key that will be used as the name of the pseudo-decision that moves the camera to the map area.
`zoom` is the zoom level for the map area, set in the amount of pixels off the ground, meaning that a lower value results in it being zoomed in more. For comparison, by default, the player can change the camera zoom between 50[1] and 3000[2].
Additionally, each of the trigger blocks defined for decisions and categories (aside from `available`) can go inside of `on_map_area`, such as `target_root_trigger`.

Scripted GUI[编辑 | 编辑源代码]

A decision category can be set to have a scripted graphical user interface container within the category, showing up below the description. The scripted GUI in question must have the decision_category context type for this to work. This is set with `scripted_gui = my_scripted_gui`. See details on the dedicated page for scripted GUI.

Category examples[编辑 | 编辑源代码]

```
POL_my_category = {
  allowed = {
    tag = POL
  }
  priority = 10
  picture = GFX_decision_category_picture
  icon = POL_category
  visible_when_empty = yes
  scripted_gui = POL_scripted_gui
}
```

```
my_map_category = {
  visible = {
    has_completed_focus = my_focus
  }
  icon = my_map
```

```
  highlight_states = {
    highlight_states_trigger = {
      is_owned_by = ROOT
      is_capital = yes
    }
  }
  on_map_area = {
    state = 123
    targets = { capital }
    zoom = 350
  }
}
```

Arguments for decisions[编辑 | 编辑源代码]

These can only be used within decisions and cannot be used in categories.

Effects on selection[编辑 | 编辑源代码]

`complete_effect` is the block of effects that gets executed immediately when the decision is selected (Starting the timer if it has one).

```
complete_effect = {
  annex_country = { target = QAT }
}
```

Decision reappearing[编辑 | 编辑源代码]

By default, a decision reappears on the very next day after being taken. In order to increase the cooldown in days, `days_re_enable = 123` can be used to put in more days in the cooldown.

In order to make the decision disappear forever upon being completed, the `fire_only_once = yes` line of code will ensure that, making the decision only be possible to fire once per country.

Decision cost[编辑 | 编辑源代码]

In order to assign a political power cost to a decision, the `cost = <int>` argument is used. The cost can be assigned a variable. As an example of a decision which is assigned a cost of 50 political power:

```
find_resources = {
  develop_infrastructure = {
    cost = 50
    available = {
      has_manpower > 500
    }
    complete_effect = {
      random_owned_state = {
        add_building_construction = { type = infrastructure level = 2 instant_build = yes }
      }
      add_manpower = -500
    }
  }
}
```

Alternatively, a decision can also take a custom localisation string as the cost. This is done with the following:

```
custom_cost_trigger = {
  <triggers>
}
custom_cost_text = <localisation key>
```

If custom_cost_trigger is fulfilled, then <localisation key> will be used for localisation, otherwise, <localisation key>_blocked will be. <localisation key>_tooltip will be used when hovering over the cost. Using the example of the following:

```
custom_cost_trigger = {
  command_power > 14
}
custom_cost_text = decision_cost_CP_15
```

The localisation file will have the following:

```
 decision_cost_CP_15:0 "£command_power  §Y15§!"
 decision_cost_CP_15_blocked:0 "£command_power  §R15§!"
 decision_cost_CP_15_tooltip:0 "It costs £command_power  §Y15§! to take the decision"
```

In order to show icons, text icons are used.
Note that a custom cost will not actually cost anything, and what you set it to cost will have to be subtracted within the `complete_effect` of the decision, preferably as a hidden effect.

Timer upon selection[编辑 | 编辑源代码]

In order to add a timer between the decision being selected and some of its effects applying, `days_remove = 123` is used to define the amount of days. If set to `-1`, the decision will never be removed by the timer running out, although additional trigger blocks can remove it.

As for defining the effects that would be executed when the timer ends, `remove_effect = { ... }` is used as an effect block. Note that `complete_effect = { ... }` is when the decision is selected starting the timer, rather than when the timer ends. In other words, this timer appears after the decision was completed and stays there until the decision is removed. Additionally, `remove_trigger = { ... }` is used to instantly remove the decision once the triggers within are met for the country, also firing the `remove_effect = { ... }` block.

Additionally, it is possible to make the decision apply modifiers during the timer's duration, with the `modifier = { ... }` providing a modifier block. Similarly to ideas, `targeted_modifier = { ... }` also exists as a way to apply targeted modifiers, in the same formatting with `tag = BHR` setting the target, such as the following example:

```
targeted_modifier = {
  tag = ENG
  attack_bonus_against = 0.1
  defense_bonus_against = -0.15
```

}

In order to make the timer cancel early without providing the effects, `visible = { ... }` does not work by default.
Instead, `cancel_trigger = { ... }` is used as a trigger block. Upon it being evaluated as true, the decision timer ends without `remove_effect = { ... }` being executed.
`cancel_if_not_visible = yes`, false by default, serves as an easy way to add visible's triggers into the `cancel_trigger = { ... }` block.
Upon the decision being cancelled, `cancel_effect = { ... }` is an effect block that gets executed. This can be useful as a way to reverse the effects applied within the complete effect.

AI[编辑 | 编辑源代码]

> *Main article: AI modding#MTTH blocks*

The chance for AI to pick a decision is decided by the `ai_will_do = { ... }` block within a decision, which is a MTTH block. **By default, a decision will never be chosen by AI,** and that block is required to make AI choose it.

As a MTTH block, the structure consists of `factor = 10` or `base = 10` to choose the initial value and `modifier = { ... }` as trigger blocks assigning `add`/`factor`/`base` in the order. For example, the following will make the decision be never chosen by AI, unless the country is at war with ITA, in which case it has 10 weight:

```
ai_will_do = {
  base = 0
  modifier = {
    add = 10
    has_war_with = ITA
  }
}
```

Warning for war[编辑 | 编辑源代码]

If your decision leads to a nation declaring war on another nation, there are several arguments that can be used to inform the targeted nation that a war is coming, as well as alert the AI to begin moving troops onto the border:
`war_with_on_remove = TAG` will make the game assume that the decision, within its `remove_effect` will declare war on the specified country, making it prepare when the timer starts.
`war_with_on_complete = TAG` will make the game assume that the decision, within its `complete_effect` will declare war on the specified country, making it prepare when the decision becomes available.

These arguments do not work for targeted decisions; see Targeted Decisions: Warning for War.

Fixed random seed[编辑 | 编辑源代码]

Decisions, by default, use a fixed random seed. What this means is that such scopes as random_list or random_owned_controlled_state will pick the same thing each time that the decision is used, making a choice when the game starts. `fixed_random_seed = no` will make sure that the random seed is dynamic, making it possible for a different outcome to happen.

Decision examples[编辑 | 编辑源代码]

QAT_category and BHR_category are assumed to already have been created within any /Hearts of Iron IV/common/decisions/categories/*.txt file.

```
QAT_category = {
  QAT_example = {
    allowed = {
      tag = QAT
    }
    icon = QAT_example    #For GFX_decision_QAT_example
    fire_only_once = yes
    days_remove = 100
    war_with_on_remove = BHR
    modifier = {
      training_time_factor = -0.3
    }
    remove_effect = {
      create_wargoal = {
        target = BHR
        type = puppet_focus_wargoal
      }
    }
    cancel_trigger = {
      OMA = {
        is_subject_of = BHR
      }
    }
    cancel_effect = {
      BHR = {
        puppet = ROOT
      }
    }
  }
}

BHR_category = {
  BHR_example = {
    allowed = {
      tag = BHR
    }
    visible = {
      has_war_with = QAT
    }
    available = {
      QAT = {
        surrender_progress > 0.5
```

```
      }
    }
    icon = GFX_decision_BHR_example
    priority = 10
    days_re_enable = 200
    custom_cost_trigger = {
      has_command_power > 14
    }
    custom_cost_text = decision_cost_CP_15
    war_with_on_complete = OMA
    fixed_random_seed = no
    complete_effect = {
      hidden_effect = {
        add_command_power = -15
      }
      annex_country = { target = QAT }
      random = {
        chance = 50
        OMA = { load_oob = "OMA_prepared" }
      }
      declare_war_on = {
        target = OMA
        type = annex_everything
      }
    }
  }
}
```

Missions[编辑 | 编辑源代码]

Missions are another type of decisions, activated when the triggers are true and requiring the player to do an action in order for the mission to have a positive outcome.

A decision is turned into a mission by adding the `days_mission_timeout = 123` line, specifying how long the mission's timeout time is. As soon as the `activation = { ... }` trigger block is met, checked daily, the mission will appear and not disappear **regardless whether the `activation = { ... }` or `visible = { ... }` blocks are met or not**. Instead, `cancel_trigger = { ... }` can be used to cancel an ongoing mission.
To reiterate, **`visible = { ... }` gets fully ignored by missions, never being checked**, however the `allowed = { ... }` block is checked at the game's start and savefile loading regardless.

The mission requires `available = { ... }` to be fulfilled, in which case `complete_effect` will be executed immediately (Or, if the mission has `selectable_mission = yes`, when the player clicks on it).

The effects executed when the timer ends, without `complete_effect = { ... }` being executed, are put within `timeout_effect = { ... }`.

By default, it's assumed that `complete_effect = { ... }` is desirable for generating the tooltip. If, instead, the player should be told to avoid `available = { ... }` from being true and pursue the mission timing out, `is_good = yes` will change the mission's tooltip accordingly.

Additionally, `war_with_on_timeout = yes` will make the game assume that the mission, within its `timeout_effect = { ... }` will declare war on the specified country, making the country's AI prepare for war as well as notifying the opposing country that there's a wargoal being justified on it, making its AI prepare as well.

As it is possible to use the `activate_mission = mission_name` effect to activate a mission bypassing the `allowed = { ... }` trigger block, for better performance with the mod it is possible to make the mission be never allowed (`always = no`) and then make sure it's only activated with that effect.

Mission example[编辑 | 编辑源代码]

```
my_mission = {
  activation = {
    has_civil_war = yes
  }
  available = {
    has_civil_war = no
    has_war = yes
  }
  cancel_trigger = {
    has_war = no
  }
  icon = mission_icon    # For GFX_decision_mission_icon
  is_good = yes
  war_with_on_timeout = SOV
  days_mission_timeout = 100
  selectable_mission = yes
  complete_effect = {
    add_ideas = my_idea
  }
  timeout_effect = {
    declare_war_on = {
      target = SOV
      type = annex_everything
    }
  }
}
```

Targeted decisions[编辑 | 编辑源代码]

In addition to regular decisions that are taken by the country towards that same country, it is possible to make a decision be targeted towards a different country or group of countries. In that case, the decision will clone itself for each country that it gets targeted towards, putting the flag of the country in the bottom right of the decision's icon. The targeted country will be possible to reference in code using FROM, while ROOT is still the country that gets the decision. **Despite what the names imply, FROM is the target of the decision rather than the country doing it.**

`fire_only_once`, in this case, will make the decision fire only once per each target country: a decision targeted towards BLR will not disappear if a decision of the same ID targered towards LIT gets completed.
Missions, as well as regular decisions, can be made targeted.

A decision becomes targeted if any way to check for a target is added within the decision:

Checking for target[编辑 | 编辑源代码]

There are several ways to limit the selection of targets. If any of these three is present in the decision, it will be marked as targeted.

The primary way to limit the selection to a select few countries is `targets = { TAG TAG }`. In that case, if you want the game to check every country with that original tag (including civil war rebellions and other types of dynamic countries), then `targets_dynamic = yes` line of code will ensure that the game will check more than the country that the country tag truly belongs to. Additionally, by default, it's impossible to target a non-existing country. The `target_non_existing = yes` argument can be used to remove that restriction.

Additionally, it is possible to use arrays to limit a selection with `target_array = array_name`, where the array must be assigned to the country. You can see a list of arrays automatically generated by the game here.

Together with any of the previous two ways or without them, `target_trigger = { ... }` is a trigger block evaluated daily for both FROM and ROOT, if `target_root_trigger` is evaluated as true.

For targeted decisions, `target_root_trigger` is also possible to use, as a trigger block that continuously checks every day if allowed was met, required to make the decision or the entire category be visible in the decision selection screen. **This checks only ROOT**, being executed before `target_trigger = { ... }` is. This exists for optimisation purposes, as it takes much less time to check the condition for one country daily than for every combination of countries.

Triggers overview[编辑 | 编辑源代码]

`allowed = { ... }` checks the country that should get this decision and only checks upon the game's start or when reloading decisions such as by loading a savegame. If false, the decision is permamently disabled.

`target_root_trigger = { ... }` checks the country the country that should get this decision, every day. If false, the decision will not appear until the next daily check. **Despite only checking one country, this still makes the decision targeted.**

`target_trigger = { ... }` checks any countries (or states) that meet `targets = { ... }` and `target_array = array_name`, if they're present. In here, ROOT (default scope) is the country that gets the decision and FROM is the potential target of the decision. This is checked daily, making the decision not appear until the next day's check if false. Putting this automatically makes the decision targeted.

`visible = { ... }` and `available = { ... }` are checked every tick. If the decision is targeted, then FROM is checked alongside ROOT, otherwise only ROOT is checked.

Warning for war[编辑 | 编辑源代码]

The regular arguments of `war_with_on_` do not work if using FROM as a target. Instead, there are alternatives for targeted decisions specifically:

`war_with_target_on_complete = yes` - Equivalent to `war_with_on_complete = FROM`
`war_with_target_on_remove = yes` - Equivalent to `war_with_on_remove = FROM`
`war_with_target_on_timeout = yes` - Equivalent to `war_with_on_timeout = FROM`

Additional note[编辑 | 编辑源代码]

Similarly to missions, the `activate_targeted_decision = { target = TAG decision = my_decision }` effect exists. If possible, it is recommended to use this effect instead of letting it automatically activate, making it never allowed. For instance, if the targeted decision activates by being at war with a country, on_war_relation_added in on_actions can be used to activate it instead of using `target_array = enemies`.

Targeted decision example[编辑 | 编辑源代码]

```
my_targeted_decision = {
  target_root_trigger = {
    has_completed_focus = my_focus
  }
  targets = { BHR QAT SAU OMA YEM IRQ SYR LEB ISR PAL }
  targets_dynamic = yes
  target_trigger = {
    FROM = {
      has_idea = my_idea
    }
  }
  icon = my_icon
  cost = 20
  war_with_target_on_complete = yes
  complete_effect = {
    create_wargoal = {
      target = FROM
      type = annex_everything
    }
  }
}
```

State targeted decisions[编辑 | 编辑源代码]

If the decision has `state_target = yes` then, instead, it'll be targeted towards a state. Targeting still works the same way, with it having FROM as a state scope instead. `targets` becomes `targets = { 123 321 }` or, in case of one state total, `targets = { state = 123 }` can be used as well.

Decisions targeted towards states will have the icon appear over the states while the decision menu is open. To prevent confusion, a map area can be added to the decision category.

Additionally, The argument `on_map_mode` will determine where the targeted decisions appear in.

`on_map_mode = map_only` will make the targeted decisions only appear on the map.
`on_map_mode = decision_view_only` will make the targeted decisions only appear in the decisions menu.
`on_map_mode = map_and_decisions_view` will make the targeted decisions appear on both the map and the decisions menu.

Example[编辑 | 编辑源代码]

```
my_state_targeted_decision = {
```

```
    target_root_trigger = {
        has_completed_focus = my_focus
    }
    target_array = GER.core_states
    target_trigger = {
        FROM = {
            is_owned_by = ROOT
        }
    }
    on_map_mode = map_and_decisions_view
    icon = my_icon
    cost = 20
    complete_effect = {
        FROM = {
            remove_core_of = GER
        }
    }
}
```

↑ `CAMERA_MIN_HEIGHT = 50.0` in Defines

↑ `CAMERA_MAX_HEIGHT = 3000.0` in Defines