Event modding - Hearts of Iron 4 Wiki

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

Events are defined in the /Hearts of Iron IV/events/ folder. There are several types of events, changing the appearance or the target of the events.

The list of event types includes `country_event`, `news_event`, `state_event`, `unit_leader_event`, and `operative_leader_event`.

In each event type, the ROOT scope refers to the country getting the event, however, the default assumed scope, also known as THIS, is not always purely the country. In state events, operative leader events, and unit leader events, the assumed scope is *both* the country and the event-specific scope type, accepting effects for either scope, sorting them at will. In case of overlap between something being possible for the country or the other scope (e.g. `add_manpower` being both a state- and country-scoped effect), the country is preferred.

Due to this confusion, it is recommended to avoid state events as they appear identical to country events (Event targets can be used within country events aimed to a specific state instead), however unit leader and operative leader events can be unavoidable due to their distinct appearance. **There is nearly no difference between event types**, with the exception of appearance, default scopes, and the fact that news event can have their pop-up disabled (However, the country will still get them). If the player country gets an event and doesn't pick a choice, the first option will get assumed by default.

When fired using an effect, the game ports over the regular event targets existing in the previous effect block to the event, as well as the ROOT scope of the block, which becomes FROM within the event. The previous effect block's FROM scopes get shifted one level down, with the effect block's FROM becoming FROM.FROM, previous FROM.FROM becoming FROM.FROM.FROM, and et cetera. Due to this, FROM is commonly called the 'sender' of the event in jargon, making FROM.FROM the sender to the sender. Although this does not always apply, as FROM.FROM can be something else entirely, e.g. an additional scope within on actions or the target of a decision. Importantly, it's what the FROM block of the effect block that fired it is, which depends on where it's fired exactly.
This does not happen if the event gets fired using an on action's random_events block: the same scoping rules as in the regular on action apply.

It is notable that a country that does not exist (i.e. does not own any states) cannot get an event in any way, even if fired via an effect. This also applies to major events: if a non-existing country gets a major event (nearly always news events) fired via an effect, nothing will appear for any country until the country in question gets its independence, upon which the event will be fired instantly.

Event creation[编辑 | 编辑源代码]

Each event is contained within a code block corresponding to the event type, such as `country_event` or `news_event`. Within the event, a mandatory line is `id`, corresponding to the event's ID, such as `id = my_event.123`.

ID rules[编辑 | 编辑源代码]

Within an event file, all events have to have an ID in the format of `<namespace>.<integer ID>`. For example, in an event `my_event.123`, the namespace will be "my_event", while the ID is "123".

A namespace must be first created before defining any events that use it. This is done with the line `add_namespace = my_event` **which must be outside of any event**. If a namespace is not defined, then the event ID will be considered a malformed token, leading it to not working in-game. The event namespace consists of word characters (e.g. letters of the English alphabet or underscores). This includes dots, so `my_event.subtopic` is a valid namespace, with events of the type of `id = my_event.subtopic.1`. The numeric ID will be taken to be everything after the last dot converted to an integer, or 0 if the conversion fails.

Internal IDs are defined for events by assigning an ID to each namespace (IDs are assigned in the order in the code, files being loaded by filename in the character order), with the first-defined namespace being assigned an ID of 10, incrementing it by 1 for each next created namespace, multiplying it by 100000, and adding the integer ID. If the ID after the namespace fails conversion to an integer, then it'll default to 0. For this reason, every event with a non-integer ID will be considered the same event, so **the ID of the event after the namespace must be an integer**.

Localisation[编辑 | 编辑源代码]

The lines `title` and `desc` are used to assign a localisation key to the event, creating its title and description depending on the current language of the game. An example line is `title = my_event.123.t` or `desc = my_event_description`. An event is required to have a title or a description unless it is hidden.

Localisation is defined in the /Hearts of Iron IV/localisation/english/ folder for the English language. It is preferable to use a new file in the folder instead of overwriting any base game files. The newly-created file will have to end with `_l_english.yml` in its filename (Note that it is a lowercase L, not an uppercase i) for it to be loaded properly. Additionally, it has to use the UTF-8-BOM text encoding. Exact details on how to change the encoding depend on the text editor used. Within the file, assuming that `l_english:` is already added as the first line, localisation can be added as such:

```
 my_event.123.t: "My event title"
 my_event_description: "My event description"
```

It is also possible to add multiple titles and descriptions to an event, making it choose one depending on conditions. This is done as the following:

```
   text = my_event.123.t.a
title = {
   trigger = {
      tag = ENG
   }
}
title = {
   text = my_event.123.t.b
}
```

The game will choose the first localisation key where the conditions are met. In this case, the event title will use the my_event.123.t.a localisation key if the country receiving the event has the tag of ENG, and every other country will have the event title use the my_event.123.t.b localisation key. `trigger` is a trigger block, requiring all of the triggers inside to be true by default. The formatting for event descriptions is the same, with `title` changed for `desc`.

Picture[编辑 | 编辑源代码]

In order to add a picture to be shown for the event, the `picture` argument is used with the name of the sprite leading to the file of the picture, such as `picture = GFX_my_sprite`.

Sprites are defined in any /Hearts of Iron IV/interface/*.gfx file, by default using eventpictures.gfx, opened within any text editor. It is recommended to create a new file in the folder instead of using a base game file in the mod for update compatibility reasons.
Within the /Hearts of Iron IV/interface/*.gfx file of your choice, the following lines can be added within the `spriteTypes = { ... }` block to define a sprite:

```
spriteType = {
   name = "GFX_my_sprite"
   texturefile = "gfx/event_pictures/my_event_picture.dds"
}
```

After creating this sprite, the file <yourmod>/gfx/event_pictures/my_event_picture.dds can be used within an event as `picture = GFX_my_sprite`

Triggering[编辑 | 编辑源代码]

In order to control when the event should fire, the `trigger` trigger block is used, requiring each condition to be fulfilled for the event to fire. This will look like the following:

```
trigger = {
  tag = GER
  has_political_power > 100
}
```

Note that the trigger is evaluated for each country individually every 20 days[1] *for the automatic firing* done without `is_triggered_only = yes`. If the event is fired in console (using the `event my_event.123` console command), the console's outputs will show which triggers were met and which weren't at the time of it being fired.

`mean_time_to_happen` can be used to change the time for the event to fire. `days` as a whole number is the base amount of days used in the calculation. `months` and `years` are also used as the base, both being whole numbers, getting multiplied by 30 and 365 respectively. Additionally, `modifier` can be used as a block with arguments of either `add` to add the amount of days, `factor` to multiply the amount of days, or `base` to change the base amount of days if the conditions are met. This looks like the following:

```
mean_time_to_happen = {
  days = 10
  years = 1
  modifier = {
    factor = 0.2      # Amount of days is multiplied by 20% for POL.
    tag = POL
  }
}
```

Once the event triggers are evaluated as true in a 20-day check, the mean time to happen is checked daily. Despite its name, it is not an arithmetic average, but a median: an event has a 50% chance to fire within the given day range and a 50% chance to fire later. Daily, the chance for an event to fire is the following, if the mean-time-to-happen's value is taken as M:
{\displaystyle 1-2^{-{\frac {1}{M}}}}
Since this check is evaluated daily for each country where the triggers are met, it is preferable to limit the amount of events with a mean time to happen to prevent slowdown.

It can be preferable to disable the event firing automatically by using `is_triggered_only = yes` - it being set to true will ensure that the event will not fire automatically and can only be triggered via an effect, such as a focus reward or a different event's option: particularly, the effects of the type country_event, news_event, and so on depending on the event's type. This can be useful for event chains. **If omitted or set to false, the event will fire automatically even if the trigger isn't defined.**
It is to be noted that `trigger` is not entirely useless in this case: it will be evaluated when the event is intended to fire, preventing it from firing if not met, which can be useful if the event is set to fire with a delay or for the random_events section in on actions. For optimisation reasons, it is preferable to make the event be triggered only if possible: on actions can commonly be used to fire the event, such as on_startup serving as a way to fire an event on a specific day, by defining a delay in days from the start date of the game.

If a event is to only fire once *total*, the `fire_only_once = yes` can be added. This will prevent the event from firing more than once in any way, be that via an effect or automatically with the mean time to happen. This means once *total*: if it gets fired for any country in any way, no other country will be able to get the event again. It will, however, still be possible to fire it in console again.

In order to make an event fire for every country, such as is done in news events, `major = yes` can be used. Note that this cannot be used in conjunction with `fire_only_once = yes`, as that makes it fire only once *total* rather than per country, meaning that it'll only fire for the country where the event is originally triggered. This will bypass the `trigger = { ... }` block for the countries that did not have this event fired originally, instead relying on the `show_major = { ... }` trigger block, if one is present. Additionally, `fire_for_sender = no`, if added, will prevent the major event from appearing for the country that it originally got fired for, via an effect or by the trigger being met.

Options[编辑 | 编辑源代码]

An event option is added with an `option = { ... }` block. An event option is an effect block, with a few extra options:

`name` decides the localisation key used for the option, such as `name = my_option_name`. It is not possible to make the option name depend on the triggers in the same way it's possible for event titles, instead, completely different event options can be used, disabling each one with a name that shouldn't be used.

`trigger = { ... }` is a trigger block, deciding when the option is visible to be picked. If the trigger is false at the time of being fired, it will not appear until the event is fired again. Additionally, for major events, `original_recipient_only = yes` can be used to ensure that only the country that fired the event has this option available, with others not having it.

`ai_chance = { ... }` is a block deciding the AI chance for event options: deciding in a proportional way which option to pick. The AI chance in event options do not have to add up to 100, as it is proportional. It is structured in a near-identical way to the mean time to happen. If unset, assumed to be 1. The probability of each option is its weight divided by the sum of all option weights. If all options have a weight of zero, the first one is chosen. The randomized choice is made by rolling a d100, so options can't have an effective non-zero probability below 1%. The choice remains consistent across reloads, based on unique game seed, in-game time, country, and unit leader.

Additional arguments[编辑 | 编辑源代码]

`immediate = { ... }` is an effect block, executed as soon as the event is fired, before an option is chosen by the player. This can also be used for AI: AI only picks an option after the event triggers are evaluated for every other country, while immediate is executed immediately, before evaluating other events. This can be used in mean-time-to-happen type major events: by making the immediate set a global flag, which is required to be unset in the event trigger, this will prevent it from being fired more than once for each country, but it is preferable to avoid mean-time-to-happen events in entirety. Note that the effect will appear in the tooltip after the event's description, so the hidden_effect flow tool can be helpful.

`timeout_days = 20` sets the amount of days that the player has to pick an option before the first option is automatically selected. This can be used to make the event be more or less urgent than default. If unset, assumes to be 13 days[2].

`hidden = yes` will make an event hidden. A hidden event does not need a title or a description. The first defined option, if one is present, will be automatically picked upon being fired. Hidden events can be useful instead of scripted effects to delay the execution of an effect block by a period of time or to utilise the FROM scope.

`minor_flavor = yes` marks the event as being a minor flavour event. This does not change its appearance or change its effects, but allows turning off the pop-up within the game's decision menu.

Effect[编辑 | 编辑源代码]

Any effect block can be used to fire an event, such as focus rewards, event options, or decision effects. This is usually paired with `is_triggered_only = yes` within the event as to disable automatic firing.

In its simplest way, this is done with the effect of `country_event = my_event.1` (or `news_event = my_event.1`), which'll instantly fire the event for the scoped country. As country and news events are the exact same thing under the surface, both shortened effects can be used for either country and news events, with there being no difference between them whatsoever.
However, more options can be added primarily for setting up the delay. Additionally, expanded versions are mandatory for state and operative leader events.

A more complex effect to fire is `country_event = { id = my_event.1 days = 100 random_days = 123 }`. This'll fire the event in 100 to 223 ({\displaystyle 100+123}) days. There are the following arguments that can go into the effect (All of them are optional with one exception):

`id = my_event.1` — The ID of the event to fire. This is mandatory as to let the game know which event to fire.

`hours = 1|days = 2|months = 3` — The lower bound on the needed time that the event will fire in. In this case, a month is treated as exactly 30 days. If multiple of these are used, the game will add them up together (e.g. the example with 1 hour, 2 days, and 3 months will fire in 92 days and 1 hour).

`random_hours = 1|random_days = 2` — This sets the upper bound on the needed time that the event will fire in. The game selects a random amount (uniform distribution) of days and hours between 0 and the set amount, including both ends, and adds them to the delay to fire the event. Similarly to above, if both are specified, the game will add them together. `random = 123` also serves as an equivalent for `random_hours = 123`.

`tooltip = my_event.1.t` — This decides what the name of the event that would fire would be displayed as in localisation. Defaults to the event's name, this may be useful if the title may change

between the effect to fire it and its actual appearance.

Example effect using several of these parameters:

```
country_event = {
  id = my_event.1
  hours = 12
  random_hours = 6
  days = 2
  tooltip = another_event.1.t
}
```

The optional delay is incredibly useful, as hidden events can be used to create a delay between an effect block's execution and the actual application of effects without the player detecting anything. This can also be used to make the events appear more "natural": news events can have a delay of a few hours (typically around 6-12) in order to simulate the time it takes for the news agencies to report on the event. Similarly can be done with other event types to simulate waiting a few hours for a diplomatic response, rather than it being unnaturally instant.
`trigger = { ... }` of the event gets checked when it would fire, meaning that it's also possible to fire the event on startup of the game with a needed delay to get it on a specific day, then use the event's trigger to simulate additional requirements.

Additionally, there are these event type-specific arguments:

`trigger_for = TAG` (Unique to state events) — The country for which the event will fire for. This is mandatory as state events are to be fired in state scope. This can also be replaced with `controller`, `owner`, `occupied`, or a dual scope that can be used as a target, such as ROOT or FROM.

`originator = TAG` (Unique to operative leader events) — The country which serves as the originator of the event (i.e. FROM). Defaults to the operative's owner if unset.

`recipient = TAG` (Unique to operative leader events) — The country which would receive the event. Defaults to the operative's owner if unset.

`set_root = TAG` (Unique to operative leader events) — Changes the scope of ROOT within the dynamic localisation of the event, without actually changing it in code.

`set_from = TAG` (Unique to operative leader events) — Changes the scope of FROM within the dynamic localisation of the event, without actually changing it in code.

`set_from_from = TAG` (Unique to operative leader events) — Changes the scope of FROM.FROM within the dynamic localisation of the event, without actually changing it in code.

Common mistakes[编辑 | 编辑源代码]

Some errors are quite common to make when beginning to make events, whether it's poor practice or if it would prevent the event from working in entirely. Some of them may be hard to notice when modding, with the event seemingly working fine, such as the error that prevents news events from being fired for more than one country.
This covers some of them, as well as the less intuitive errors in the log.

Unlogged errors[编辑 | 编辑源代码]

Leaving an event as triggered only when it's to be fired automatically (*Event never fires*) – `is_triggered_only = yes` disables the automatic firing of the event, instead enforcing using the effect to do so. Therefore, if it is left in within an event intended to fire automatically, the event will never do so.

> Note that `trigger = { ... }` can still co-exist with `is_triggered_only = yes`, so an event with both `is_triggered_only = yes` and `trigger = { ... }` may still be correct. Valid usages of them at the same time include the event being triggered via an on action's `random_events` or the effect firing it has a delay (where the trigger would check if when the event is to be received), along others.

展开Collapsed example event with this issue and a correction to it

Not checking the country in the trigger for country-specific auto-triggered events (*Event fires for the wrong country/never fires*) – The events are not assigned to countries in any way (namespaces and filenames serve a purely organisational purpose), and each event trigger is checked for each country in order specified in the tag list.

> In the provided example, the event requires ITA to have more than 123 political power, upon which the country receiving the event would annex AUS. However, once ITA has that much, this trigger would be true *regardless* of where it's checked. The first country in the taglist by default is GER, and so it'll be the first country where the triggers are checked. In practice, this event will result in GER annexing AUS rather than ITA.
> In the correction, a change is made: first it checks that the country that would receive the event is ITA, and only then then it checks that it has more than enough political power. This makes sure that no other countries can receive this event. Specifying the tag is unnecessary if the trigger itself already implies a certain tag (e.g. `has_completed_focus` with a tag-specific focus tree), but is needed otherwise.

展开Collapsed example event with this issue and a correction to it

Unnecessarily using auto-triggered events instead of ones that are triggered only (*Poor practice/optimisation*) – This is more of a poor practice than an error. In general, if an event's condition can be triggered with an effect, it should be.

> The example is the most obvious way of doing this: a has_completed_focus check instead of firing it directly in the focus. However, other such cases can occur, e.g. when a war starts between two countries, when a state gets occupied, or for firing one on a specific date. It's best practice to check on actions before creating an automatically-triggered event to see if they can be made to replicate.
> Firing it via an effect has a purpose of being instant instead of having to wait up to 20 days. If so desired, a delay of a few hours can be added to make it appear more natural to the player. Additionally, it serves as a way to optimise the modification, as this reduces the amount of trigger checks repeatedly done. Events with a large mean-time-to-happen are particularly awful for performance and can be replaced with an effect block firing one with a large delay created with random_days within the effect in some cases.

展开Collapsed example event with this issue and a correction to it

Tight bounds on date triggers (*Event never fires*)/**Auto-triggered event intended to be fired at a specific date** (*Event fires later than intended*) – The `trigger = { ... }` block is checked every 20 days by default, and this isn't possible to change for just one event in particular. If the date triggers are set with tight upper and lower bounds (e.g. `date > 1936.1.1` and `date < 1936.1.3`), it's very likely that the event will never fire, as this will not force the game to check the trigger at that date, but just prevent it from firing the event if the range is never checked, as the game doesn't see into the future and cannot predict that the trigger will be true or false at some point. Similarly, just placing a `date > 1936.1.1` will not ensure the event will be fired at exactly the second of January, but it may be anywhere between the 2nd and 21st (though it will be the same day on each reset).

> In order to fire an event at a certain date, it's best to fire it on startup with the needed delay, setting the event to never fire by itself with `is_triggered_only = yes` and optionally adding additional prerequisites for it to appear within `trigger = { ... }`, which would be checked at the moment the event is intended to appear. For the calculation of the amount of days to be correct, leap days must be ignored as the game doesn't contain them.
> Any effect block executed before or during startup can be used. An example using the on_startup on action exists further down the article.

Setting a news event to fire only once or not setting one as major (*News event only fires for one country*) – An event requires `major = yes` in order to appear for every country. News events are purely a reskin of country events and are not set to fire for every country by default, so this line is mandatory. Alongside that, events that fire only once don't appear more than once *globally* rather than per country. The event appearing for more than one country counts as it firing once again, so setting an event to fire only once will lead to only one country getting the news event instead of every one as intended.

展开Collapsed example events with this issue

Unintuitive logged errors[编辑 | 编辑源代码**]**

Event is triggered only, but does not have a 1 base-factor. – This occurs when there are contradictory arguments within an event about whether it's allowed to be fired automatically or if it can only be fired manually. In particular, `mean_time_to_happen = { ... }` only has an effect when the event can only be fired automatically. However, if the event is triggered only, it cannot be fired automatically, resulting in the error being created.

展开Collapsed example event with this issue

Event is set to trigger every day. – This occurs when all of the following is true for the event:
The event is possible to be fired automatically. In other words, `is_triggered_only = yes` is **not** present in the event.

The event has a mean time to happen of 1 or less days. If it is omitted, it counts as 1 day.

The event can fire more than once. In other words, `fire_only_once = yes` is **not** present in the event.

> This error warns the player that the event may fire every day once the `trigger = { ... }` evaluates as true in a check that happens every 20 days. In order to remove the error, either of the three necessary clauses can be made to be not true for the event. For example, in a lot of cases, it is possible to make the event not be fired automatically and use an effect block to fire it instead, such as by using on actions.

展开Collapsed example event with this issue

Malformed token: event_id.123, near line on a line specifying the event ID – This occurs if the event namespace was not added as needed.

Failed to create id 12300000 50. Already exists in game. This might crash the game. Reverse id lookup: id 12300000 = my_namespace.0 – Note that this is the exact same error split into two instead of being two separate errors as it might seem on the first glance. This means that the internal event ID is used by 2 or more events. There are the following reasons for this error to appear:
Putting 2 events with the exact same ID by error – `id = my_namespace.0` is included in two events at once. This is self-explanatory.

Using non-integers as the numeric ID after the namespace – One event has `id = my_namespace.abc`, the other has `id = my_namespace.cba`. Due to how the game generates internal IDs, a non-numeric ID is not supported, always becoming the number 0. As such, these are the exact same ID, even if they appear different.

Using a numeric ID not smaller than 100000 – Two events across different namespaces got assigned the same internal IDs. By the virtue of how the game generates internal IDs, each namespace is assigned 100000 numeric IDs, from 0 to 99999. Anything larger than that will start encroaching to other namespaces' IDs. The game fully allows numbers this large as the numeric IDs, so the event might work, but the duplicate internal ID means that there is a pair of events that are treated as the same event, meaning one of them will fire the other one instead.

> Since the reverse id lookup is not always provided, the way to tell if this is event-related is the second number: `50` signifies that it's event-related, while a different number means a different database entry, e.g. `54` means country leader IDs and `55` means unit leader IDs, the numeric legacy IDs which are unneeded due to the 1.11-introduced character system

展开Collapsed example event file with this issue

Event file example[编辑 | 编辑源代码]

```
add_namespace = my_event
add_namespace = my_hidden_event

country_event = {
  id = my_event.1
  title = my_event.1.t
  desc = my_event.1.desc

  is_triggered_only = yes

  option = {
    name = my_event.1.a
    add_political_power = 100
  }
}

add_namespace = my_news_event
news_event = {
  id = my_news_event.1
  title = {
    text = my_news_event.1.t.a
    trigger = {
      tag = POL
    }
  }
  title = {
    text = my_event.1.t
  }
  desc = {
    text = my_news_event.1.desc.a
    trigger = {
      tag = POL
    }
  }
  desc = {
    text = my_event.1.desc
  }

  picture = GFX_my_news_event_picture

  is_triggered_only = yes
  major = yes

  option = {
    name = my_news_event.1.a
```

```
    trigger = {
      tag = POL
    }
  }

  option = {
    name = my_news_event.1.b
    trigger = {
      NOT = { tag = POL }
    }
  }

  option = {
    name = my_news_event.1.c
    original_recipient_only = yes
  }
}

country_event = {
  id = my_hidden_event.1

  trigger = {
    has_country_flag = event_happened
    country_exists = BHR
  }

  mean_time_to_happen = {
    days = 10
    months = 2
    years = 1
    modifier = {
      base = 300
      country_exists = QAT
    }
    modifier = {
      add = 10
      country_exists = OMA
    }
  }

  fire_only_once = yes
  hidden = yes

  immediate = {
    random_country = {
      limit = {
        is_neighbor_of = BHR
      }
      annex_country = {
        target = BHR
        transfer_troops = yes
      }
    }
  }
}

state_event = {
  id = my_event.2
  title = my_event.2.t
  desc = my_event.2.desc
  picture = GFX_my_event_picture

  trigger = {
    ROOT = {
      has_country_flag = fire_this_event
    }
  }
  is_triggered_only = yes

  option = {
    name = my_event.2.a
    transfer_state_to = ROOT
  }

  option = {
    name = my_event.2.b
    ai_chance = {
      base = 0      # Never pick this option.
    }
    transfer_state_to = FROM
```

```
  }
}
```

Integration with on actions[编辑 | 编辑源代码**]**

See also: On actions

These events are the primary types to trigger via on_actions:

```
add_namespace = on_action_events
news_event = {     # City capture news event
  id = on_action_events.1
  title = on_action_events.1.t    # Fall of Giza
  desc = on_action_events.1.desc

  is_triggered_only = yes
  major = yes

  option = {
    trigger = {
      OR = {
        tag = EGY
        is_in_faction_with = EGY
        is_subject_of = EGY
      }
    }
    name = on_action_events.1.a
  }

  option = {
    trigger = {
      NOT = {
        tag = EGY
        is_in_faction_with = EGY
        is_subject_of = EGY
      }
    }
    name = on_action_events.1.b
  }
}

country_event = {      # Fired on a specific day if circumstances are met
  id = on_action_events.2
  title = on_action_events.2.t
  desc = on_action_events.2.desc

  is_triggered_only = yes                 # Prevents from firing automatically.
  trigger = {
    has_completed_focus = BHR_focus_name    # If the focus isn't completed, will never fire.
  }

  option = {
    name = on_action_events.2
  }
}

country_event = {      # Other types of on_actions
  id = on_action_events.3 # In this case, a prompt on annexing a country with an option to release it.
  title = on_action_events.3.t
  desc = on_action_events.3.desc

  is_triggered_only = yes                 # Prevents from firing automatically.

  trigger = {     # If all core states of FROM are cored or claimed by ROOT, should never appear.
    NOT = {     # Triggered within on_annex's random_events = { ... }, so has the same FROM as the on_action
      any_state = {
        NOT = {
          is_core_of = ROOT
          is_claimed_by = ROOT
        }
        is_core_of = FROM
      }
    }
    FROM = {
      NOT = {
        tag = GER      # Has separate event
      }
    }
  }

  option = {
```

```
    name = on_action_events.3.a   # "Release [FROM.GetName] as puppet"
    every_owned_state = {
      limit = {
        is_core_of = FROM
        NOT = {
          is_core_of = ROOT
          is_claimed_by = ROOT
        }
      }
      transfer_state_to = FROM
    }
    if = {
      limit = {
        has_dlc = "Together for Victory"
      }
      set_autonomy = {
        target = FROM
        autonomy_state = autonomy_integrated_puppet
      }
    }
    else = {
      puppet = FROM
    }
  }

  option = {
    name = on_action_events.3.b # "Don't release [FROM.GetName]"
    add_stability = -0.1
    add_war_support = -0.1
  }
}
```

In order to fire this, code has to be created within any /Hearts of Iron IV/common/on_actions/*.txt file, sometimes paired with boolean flags to prevent them from being fired more than once if fire_only_once is impossible. In the above example, this would be used:

```
on_actions = {
  on_state_control_changed = {
    effect = {
      if = {
        limit = {
          FROM.FROM = {
            state = 999 # Custom state ID. Will crash the game if doesn't exist.
          }
          NOT = {
            has_global_flag = giza_fall # To prevent from firing twice.
          }                             # Due to 'major = yes', a fire_only_once will NOT work
        }
        news_event = { id = on_action_events.1 hours = 6 random_hours = 3 } # Fires in 6-9 hours to feel more natural.
      }
    }
  }
  on_startup = {
    effect = {
      BHR = {
        country_event = {
          id = on_action_events.2
          days = 357
          random_days = 7     # Fires in the last week of 1936, assuming default start date.
        }
      }
    }
  }
  on_annex = {
    random_events = {
      1 = on_action_events.3
    }
  }
}
```

References[编辑 | 编辑源代码]

↑ NDefines.NCountry.EVENT_PROCESS_OFFSET = 20 in Defines

↑ NDefines.NGame.EVENT_TIMEOUT_DEFAULT = 13 in Defines

Note that when editing defines, it is far preferable to use an override file than copying over the entire file, as defines are edited commonly even in 'minor' updates, which can cause crashes when the game updates.