# Triggers

Triggers for automatic flow invocation v1.26

Sirenia

September 23, 2020

# Contents

A trigger is a mechanism by which an external event may cause a flow to be invoked.

When a flow is invoked with a trigger its `Inputs` will contain two extra pieces of information:

- `[eu.sirenia]Manatee.In.Trigger` will contain the *type* of the trigger
- `[eu.sirenia]Manatee.In.TriggerParams` will contain any parameters the trigger may provide (see below for examples)

## 1  Adding a trigger to a flow

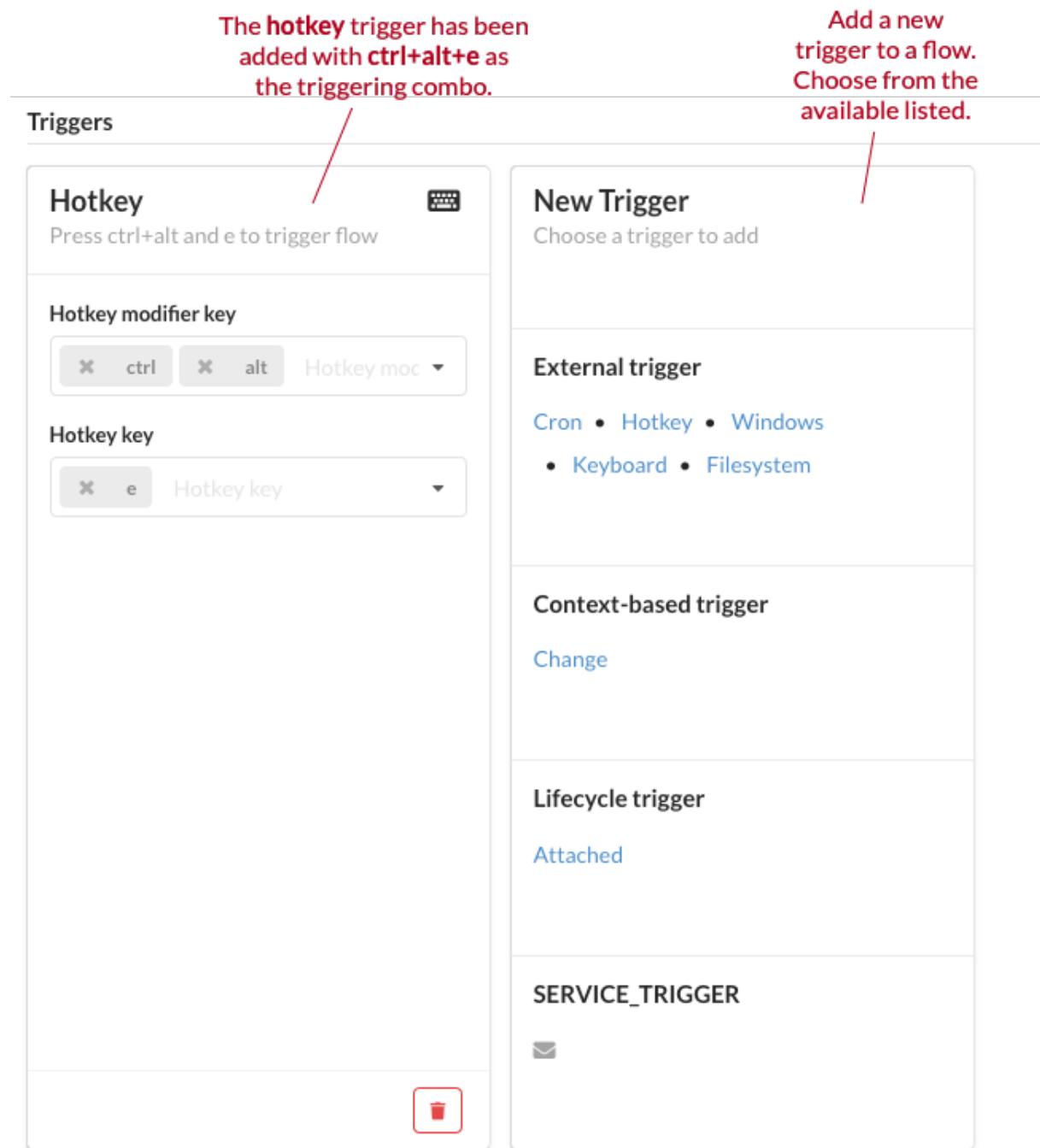A trigger can be added to a flow in the *Settings* tab for a flow.

**Figure 1:** Adding a trigger

The trigger will activate on all Manatees on which it is possible to run the flow (controlled via the *groups* of the flow and the application that contains the flow).

## 2  External triggers

External triggers are all triggers that are external to the Manatee but are still events on the *local* machine.

### 2.1  Window

The *window* trigger can launch a flow based on whether a matching window is shown or hidden.

#### 2.1.1  Configurable parameters

- `Event` which window-event to use ('shown' or 'hidden')
- `Title` the title which the window must have
- `Class` the class of the window frame
- `CloseWindow` whether or not to immediately close the detected window

#### 2.1.2  Parameters to flow

The *trigger type* is WINDOW.

- `title` the actual title of the detected window
- `hwnd` the window handle
- `kind` the kind of event (window 'shown' or 'hidden')
- `className` the class of the window frame
- `closeWindow` whether or not the window was closed

The parameters are serialized as a JSON object.

### 2.2  File-system

The *filesystem* trigger can monitor a directory for files that are either added, removed or changed.

#### 2.2.1  Configurable parameters

- `Directory` the directory to monitor
- `Match` a regular expression which filters the files that will be used to trigger the flow

### 2.2.2 Parameters to flow

The *trigger type* is `FILESYSTEM`.

- `file` the acual file that caused the trigger to fire
- `eventType` the type of event ('CREATED', 'CHANGED', 'RENAMED' or 'DELETED')

The parameters are serialized as a JSON object.

## 2.3 Cron

The *cron* trigger can launch a flow on a certain time-schedule.

### 2.3.1 Configurable parameters

The trigger is configured with a cron string which specifies a schedule with which to fire the flow. The UI in Cuesta tries to be helpful in explaining when a trigger will fire as you type in the components of the cron string.

### 2.3.2 Parameters to flow

The *trigger type* is CRON.

- `cronString` the cron string which triggered the flow

The parameters are serialized as a JSON object.

## 2.4 Hotkey

The *hotkey* trigger can launch a flow when the user hits a matching keyboard combo.

A valid combo must include at least one *modifier key* and one other *key*, e.g. `ctrl+r` is valid, while `r` is not.

### 2.4.1 Configurable parameters

- `modifierKeys` a set of modifier keys the user must press
- `keys` a set of keys the user must press while the *modifier keys* are pressed

### 2.4.2 Parameters to flow

The *trigger type* is HOTKEY.

- `modifierKeys` the set of modifier keys the user pressed
- `keys` the set of keys the user pressed

## 2.5 Keyboard

The *keyboard* trigger listens for fast keyboard input matching a given regular expression. The match-value of the first group in the regex is given as input to the flow. If you have any subgroups in your trigger regular expression then you need to wrap the whole regular expression in paranthesis to form the first group. E.g. you're using this trigger for keyboard wedge configured barcode scanners and your raw input is something like: `{OEM1}d01029284{NUMPAD0}{NUMPAD8}21982798{NUMPAD0}`. In order to be resistant to keyboard layout changes you can then construct a regular expression:

`.*d\d{8}(\{[^\}].+\})+\d{8}(\{[^\}].+\})+`

But since the first group will match the `{NUMPADN}` elements, then that is the input given to the flow. This can then be fixed by adding an initial group:

`.*d(\d{8}(\{[^\}].+\})+\d{8}(\{[^\}].+\})+)`

This trigger is most useful with devices such as scanners that can be configured to act as keyboards. The trigger will hold the input characters in a buffer for a short (while) and trigger a flow if the input buffer contains a matching sequence of characters. The input itself is not blocked and will end where the window focus is.

### 2.5.1 Configurable parameters

- `match` the regular expression which the input buffer must match
- `lookbackMs` the number of milliseconds the input buffer will contain the characters

### 2.5.2 Parameters to flow

The *trigger type* is KEYBOARD.

- `matchValue` the value of the matching string, either the first group in the regular expression or the complete match if it does not contain any groups

# 3  Context triggers

The *trigger type* is CONTEXT.

Context triggers are triggers which will fire a flow once the shared context matches the preconditions for the trigger.

## 3.1  Change

Run a flow whenever a subject in the shared context changes a value matching the configured properties of the trigger.

### 3.1.1  Configurable parameters

- `subject` the subject to match
- `oldValue` the old value to match (regular expression, default is empty which matches all values)
- `newValue` the new value to match (regular expression, default is empty which matches all values)

### 3.1.2  Parameters to flow

The *trigger type* is CONTEXT.

- `subject` the subject of the matching context item
- `oldValue` the previous value of the context item
- `newValue` the new value of the context item

# 4  Lifecycle triggers

Lifecycle triggers are coupled with the lifecycle of the application.

## 4.1  Attached

The trigger will fire as soon as the application is attached to a session. It can be used as a kind of initializer. It does not have any configurable parameters or provides any meaningful input to a flow except the type of the event; LIFECYCLE and the param ATTACHED.

# 5 Triggers from services

## 5.1 Mail

The *mail* trigger will trigger a flow whenever one or more mails matching the configured preconditons are received. It must be configured with the mail service (which in turn contains server connection details).

Only unread mails can trigger a flow and as soon as a trigger runs it will mark the mail as read. This ensures that the same flow (on the same machine) does not run multiple times on the same mail. In general the same principle applies across machines, but it is not guaranteed.

### 5.1.1 Configurable parameters

- `service` the mail service to connect to
- `fromAddress` the address to match with the sender of the mail (regular expression)
- `toAddress` the address to match with the recipient of the mail (regular expression)
- `ccAddress` the address to match with the (cc) recipient of the mail (regular expression)
- `subject` the subject to match (regular expression)
- `body` the body to match (regular expression)
- `attachments` the attachments to match (regular expression)

### 5.1.2 Parameters to flow

- `from` the senders address
- `to` the recipient address
- `cc` the cc address
- `subject` the subject
- `body` the body
- `attachments` the attachments

## 5.2 Serial port

The *serial port* trigger will trigger a flow whenever a message is received from the device, which matches the trigger message pattern.

### 5.2.1  Configurable parameters

- `Serial port` the port to monitor for messages
- `Message pattern` the pattern to match incoming messages against (regular expression). Note that this trigger is currently limited to matching inbound data interpreted as text according to the encoding selected in the chosen serial port service. For devices with binary protocols, this may make the regular expressions a bit harder to write. To match the byte sequence `0x02 0x06 0x03`, the regular expression `\x02\x06\x03` can be used. For binary protocol devices, the encoding selector in the serial port service should be left empty.

### 5.2.2  Parameters to flow

- `message` the matched message as a string

### 5.2.3  Previous messages

The serial port service keeps track of the most recently received messages from the device. Through this mechanism a flow can inspect the messages received prior to the triggering message. Details and examples can be seen in the service document.

---

See the service document for a detailed list of services.