Automation trigger variables

Automations support templating in the same way as scripts do. In addition to the Home Assistant template extensions available to scripts, the `trigger` and `this` template variables are available.

The template variable `this` is also available when evaluating any `trigger_variables` declared in the configuration.

Available this data

The variable `this` is the state object of the automation at the moment of triggering the actions. State objects also contain context data which can be used to identify the user that caused a script or automation to execute. Note that `this` will not change while executing the actions.

Available trigger data

The variable `trigger` is an object that contains details about which trigger triggered the automation.

Templates can use the data to modify the actions performed by the automation or displayed in a message. For example, you could create an automation that multiple sensors can trigger and then use the sensor's location to specify a light to activate; or you could send a notification containing the friendly name of the sensor that triggered it.

Each trigger platform can include additional data specific to that platform.

ALL

Triggers from all platforms will include the following data.

Template variable	Data
`trigger.id`	The `id` of the trigger.
`trigger.idx`	Index of the trigger. (The first trigger idx is `0`.)

CALENDAR

These are the properties available for a Calendar trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `calendar`
`trigger.event`	The trigger event type, either `start` or `end`
`trigger.calendar_event`	The calendar event object matched.
`trigger.calendar_event.summary`	The title or summary of the calendar event.
`trigger.calendar_event.start`	String representation of the start date or date time of the calendar event e.g. `2022-04-10`, or `2022-04-10 11:30:00-07:00`
`trigger.calendar_event.end`	String representation of the end time of date time the calendar event in UTC e.g. `2022-04-11`, or `2022-04-10 11:45:00-07:00`
`trigger.calendar_event.all_day`	Indicates the event spans the entire day.
`trigger.calendar_event.description`	A detailed description of the calendar event, if available.
`trigger.calendar_event.location`	Location information for the calendar event, if available.
`trigger.offset`	Timedelta object with offset to the event, if any

DEVICE

These are the properties available for a Device trigger.

Inherites template variables from event or state template based on the type of trigger selected for the device.

Template variable	Data
`trigger.platform`	Hardcoded: `device`.

EVENT

These are the properties available for a Event trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `event`.
`trigger.event`	Event object that matched.
`trigger.event.event_type`	Event type.
`trigger.event.data`	Optional event data.

MQTT

These are the properties available for a MQTT trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `mqtt`.
`trigger.topic`	Topic that received payload.
`trigger.payload`	Payload.
`trigger.payload_json`	Dictionary of the JSON parsed payload.
`trigger.qos`	QOS of payload.

NUMERIC STATE

These are the properties available for a numeric state trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `numeric_state`
`trigger.entity_id`	Entity ID that we observe.
`trigger.below`	The below threshold, if any.
`trigger.above`	The above threshold, if any.
`trigger.from_state`	The previous state object of the entity.
`trigger.to_state`	The new state object that triggered trigger.
`trigger.for`	Timedelta object how long state has met above/below criteria, if any.

SENTENCE

These are the properties available for a Sentence trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `conversation`
`trigger.sentence`	Text of the sentence that was matched
`trigger.slots`	Object with matched slot values
`trigger.details`	Object with matched slot details by name, such as wildcards. Each detail contains: • `name` - name of the slot • `text` - matched text • `value` - output value (see lists)

STATE

These are the properties available for a State trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `state`
`trigger.entity_id`	Entity ID that we observe.
`trigger.from_state`	The previous state object of the entity.
`trigger.to_state`	The new state object that triggered trigger.
`trigger.for`	Timedelta object how long state has been to state, if any.

SUN

These are the properties available for a Sun trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `sun`
`trigger.event`	The event that just happened: `sunset` or `sunrise`.
`trigger.offset`	Timedelta object with offset to the event, if any.

TEMPLATE

These are the properties available for a Template trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `template`
`trigger.entity_id`	Entity ID that caused change.

Template variable	Data
`trigger.from_state`	Previous state object of entity that caused change.
`trigger.to_state`	New state object of entity that caused template to change.
`trigger.for`	Timedelta object how long state has been to state, if any.

TIME

These are the properties available for a Time trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `time`
`trigger.now`	DateTime object that triggered the time trigger.

TIME PATTERN

These are the properties available for a time pattern trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `time_pattern`
`trigger.now`	DateTime object that triggered the time_pattern trigger.

PERSISTENT NOTIFICATION

These properties are available for a persistent notification trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `persistent_notification`
`trigger.update_type`	Type of persistent notification update `added`, `removed`, `current`, or `updated`.
`trigger.notification`	Notification object that triggered the persistent notification trigger.
`trigger.notification.notification_id`	The notification ID
`trigger.notification.title`	Title of the notification
`trigger.notification.message`	Message of the notification
`trigger.notification.created_at`	DateTime object indicating when the notification was created.

WEBHOOK

These are the properties available for a Webhook trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `webhook`
`trigger.webhook_id`	The webhook ID that was triggered.
`trigger.json`	The JSON data of the request (if it had a JSON content type) as a mapping.
`trigger.data`	The form data of the request (if it had a form data content type).
`trigger.query`	The URL query parameters of the request (if provided).

ZONE

These are the properties available for a Zone trigger.

Template variable	Data
`trigger.platform`	Hardcoded: `zone`
`trigger.entity_id`	Entity ID that we are observing.
`trigger.from_state`	Previous state object of the entity.
`trigger.to_state`	New state object of the entity.
`trigger.zone`	State object of zone
`trigger.event`	Event that trigger observed: `enter` or `leave`.

Examples

```yaml
# Example configuration.yaml entries
automation:
  trigger:
    - platform: state
      entity_id: device_tracker.paulus
      id: paulus_device
  action:
    - service: notify.notify
      data:
        message: >
          Paulus just changed from {{ trigger.from_state.state }}
          to {{ trigger.to_state.state }}

          This was triggered by {{ trigger.id }}

automation 2:
  trigger:
    - platform: mqtt
      topic: "/notify/+"
  action:
    service: >
      notify.{{ trigger.topic.split('/')[-1] }}
    data:
      message: "{{ trigger.payload }}"

automation 3:
  trigger:
    # Multiple entities for which you want to perform the same action.
    - platform: state
      entity_id:
```

```
        - light.bedroom_closet
        - light.kiddos_closet
        - light.linen_closet
      to: "on"
      # Trigger when someone leaves one of those lights on for 10 minutes.
      for: "00:10:00"
  action:
    - service: light.turn_off
      target:
        # Turn off whichever entity triggered the automation.
        entity_id: "{{ trigger.entity_id }}"

automation 4:
  trigger:
    # When an NFC tag is scanned by Home Assistant...
    - platform: event
      event_type: tag_scanned
      # ...By certain people
      context:
        user_id:
          - 06cbf6deafc54cf0b2ffa49552a396ba
          - 2df8a2a6e0be4d5d962aad2d39ed4c9c
  condition:
    # Check NFC tag (ID) is the one by the front door
    - condition: template
      value_template: "{{ trigger.event.data.tag_id == '8b6d6755-b4d5-4c23-818b-
cf224d221ab7'}}"
  action:
    # Turn off various lights
    - service: light.turn_off
      target:
        entity_id:
          - light.kitchen
          - light.bedroom
          - light.living_room
```