AI-01791 Home Assistant automation.yaml

Owner	Ⓡ Rigel Arcayan
Tags	Problem
Created time	@July 2, 2025 5:06 AM

Explain Home Assistant automation.yaml. Are all devices code residing in one yaml?

Great question, Rigel. Here's a breakdown of how `automations.yaml` works in Home Assistant and how device code is structured:

🧠 What Is `automations.yaml`?

In Home Assistant, `automations.yaml` is a dedicated file where **automation rules** are stored. These rules define how your smart home behaves based on triggers, conditions, and actions.

Each automation follows this basic structure:

```
- id: unique_automation_id
  alias: "Descriptive Name"
  trigger:
    ...
  condition:
    ...
  action:
    ...
```

🗂️ Are All Devices in One YAML File?

Not exactly. Here's how it works:

🔷 1. Device Definitions

- Devices themselves (like Zigbee sensors, switches, etc.) are **not defined** in `automations.yaml`.
- Devices are discovered and managed through **integrations** and stored in `.storage/core.device_registry` (not user-editable).
- What you use in `automations.yaml` are **entity IDs** (like `sensor.living_room_temp`) that are linked to those devices.

🔷 2. Automation Code

- All automations can be stored in `automations.yaml`, but:
  - You can **split automations** into multiple files using `!include` or `!include_dir_merge_list` in `configuration.yaml`:
    or
    This lets you organize automations by room, device type, or function.

    ```
    automation: !include automations.yaml
    ```

    ```
    automation: !include_dir_merge_list automations/
    ```

🧩 Best Practices

- ✅ **Use `entity_id` instead of `device_id`** in automations. It makes your YAML more portable and easier to maintain if you replace devices.
- ✅ A**Split automations** into multiple files if your setup grows large.
- ✅ Use the **UI editor** for simple automations and switch to YAML for advanced logic or templating.

🛠️ Example Automation (YAML)

```
- id: turn_on_lights_at_sunset
  alias: "Turn on lights at sunset"
  trigger:
```

```
    - platform: sun
      event: sunset
  condition:
    - condition: state
      entity_id: person.rigel
      state: "home"
  action:
    - service: light.turn_on
      target:
        entity_id: light.living_room
```

Would you like help organizing your automations into multiple files or converting some UI-created automations into YAML?

Absolutely, Rigel! Here are some **advanced Home Assistant automation examples** that go beyond simple triggers and showcase the platform's full potential:

🔄 1. Thermostat + Door Sensor + Time-Based Logic

This automation controls an AC plug based on multiple conditions:

- AC turns on at 8 PM.
- Mirrors thermostat state (cooling = on, idle = off) between 8 PM and 7 AM.
- Turns off if the door is open for 20 minutes.

```
- alias: "Bedroom AC Control"
  trigger:
    - platform: time
      at: "20:00:00"
    - platform: time
      at: "07:00:00"
    - platform: state
      entity_id: climate.thermostat
      attribute: hvac_action
    - platform: state
```

```
      entity_id: binary_sensor.sliding_door
      to: "on"
      for: "00:20:00"
  condition: []
  action:
    - choose:
        - conditions:
            - condition: time
              after: "20:00:00"
              before: "07:00:00"
            - condition: template
              value_template: "{{ state_attr('climate.thermostat', 'hvac_action') == 'cooling' }}"
          sequence:
            - service: switch.turn_on
              target:
                entity_id: switch.ac_plug
        - conditions:
            - condition: template
              value_template: "{{ is_state('binary_sensor.sliding_door', 'on') }}"
          sequence:
            - service: switch.turn_off
              target:
                entity_id: switch.ac_plug
        - conditions:
            - condition: template
              value_template: "{{ state_attr('climate.thermostat', 'hvac_action') == 'idle' }}"
          sequence:
            - service: switch.turn_off
              target:
                entity_id: switch.ac_plug
```

🌙 2. Night Mode Abstraction Using Helpers

Instead of repeating complex logic in every automation, use a helper (like `input_boolean.night_mode`) to centralize logic.

Night Mode Enable:

```
- alias: "Enable Night Mode"
  trigger:
    - platform: time
      at: "22:00:00"
  condition:
    - condition: state
      entity_id: person.rigel
      state: "home"
  action:
    - service: input_boolean.turn_on
      target:
        entity_id: input_boolean.night_mode
```

Night Mode Usage:

```
- alias: "Dim Lights When Motion Detected at Night"
  trigger:
    - platform: state
      entity_id: binary_sensor.hallway_motion
      to: "on"
  condition:
    - condition: state
      entity_id: input_boolean.night_mode
      state: "on"
  action:
    - service: light.turn_on
      data:
        entity_id: light.hallway
        brightness: 50
```

🧠 3. Presence-Based HVAC Optimization

Use multiple presence sensors and temperature readings to optimize HVAC usage:

```
- alias: "Smart HVAC Control"
  trigger:
    - platform: numeric_state
      entity_id: sensor.living_room_temp
      above: 28
  condition:
    - condition: state
      entity_id: group.family
      state: "home"
    - condition: time
      after: "09:00:00"
      before: "18:00:00"
  action:
    - service: climate.set_hvac_mode
      data:
        entity_id: climate.living_room
        hvac_mode: "cool"
```

📦 Want More?

You can explore full examples and community contributions here:

- Home Assistant Community: Complex Automation Thread
- [AGitHub: Home Assistant Automation Examples](https://github.com/tomwaldnz/HomeAssistantExamples)
- [BAdvanced Automation Guide](https://sensorpoweredhome.com/advanced-home-assistant-customization-automation-guide/)
- - C

Would you like me to help you build one of these for your own setup—maybe something like a smart energy saver or a security routine?

Show a Sonoff Zigbee Motion Sensor SNB-06P complex example

Thanks for waiting, Rigel! The **Sonoff SNZB-06P** is a **Zigbee human presence sensor** that uses **5.8GHz microwave radar** to detect **both motion and static presence**—even if someone is sitting still or asleep A. It’s more advanced than typical PIR motion sensors like the SNZB-03.

Here’s a **complex Home Assistant automation** example that takes full advantage of its capabilities:

🧠 Scenario: Smart Office Lighting with Presence, Brightness, and Timeout Logic

🎯 Goals:

- Turn on lights when presence is detected **and** it's dark.
- Keep lights on as long as presence is detected—even if the person is still.
- Turn off lights after 5 minutes of no presence.
- Use **illumination sensor** to avoid turning on lights during the day.

🧩 Entities Used:

- `binary_sensor.office_presence` → from SNZB-06P (occupancy)
- `sensor.office_illumination` → from SNZB-06P (values: `dim`, `bright`)
- `light.office_ceiling`
- `script.office_light_off_delay`
- `input_boolean.office_presence_active` (helper)

🛠️ Step 1: Script to Turn Off Light After Delay

```
script:
  office_light_off_delay:
    alias: "Turn off office light after delay"
    sequence:
      - delay: "00:05:00"
      - service: light.turn_off
        target:
          entity_id: light.office_ceiling
```

🛠️ Step 2: Automation – Presence Detected

```
- alias: "Office Presence Detected"
  trigger:
    - platform: state
      entity_id: binary_sensor.office_presence
      to: "on"
  condition:
    - condition: or
      conditions:
        - condition: state
          entity_id: sensor.office_illumination
          state: "dim"
        - condition: template
          value_template: "{{ states('sensor.office_illumination') | lower == 'dim' }}"
  action:
    - service: input_boolean.turn_on
      target:
        entity_id: input_boolean.office_presence_active
    - service: script.turn_off
      target:
        entity_id: script.office_light_off_delay
    - service: light.turn_on
      data:
        entity_id: light.office_ceiling
        brightness: 200
```

🛠️ Step 3: Automation – No Presence

```
- alias: "Office No Presence"
  trigger:
    - platform: state
      entity_id: binary_sensor.office_presence
      to: "off"
      for: "00:00:30"
  condition:
    - condition: state
      entity_id: input_boolean.office_presence_active
      state: "on"
  action:
    - service: script.turn_on
      target:
        entity_id: script.office_light_off_delay
    - service: input_boolean.turn_off
      target:
        entity_id: input_boolean.office_presence_active
```

🧠 Why This Is "Complex"

- Uses **radar-based presence** instead of motion.
- Integrates **ambient light sensing** to avoid unnecessary lighting.
- Uses **helper + script** to manage delayed turn-off and cancelation.
- Works even if the person is **sitting still** (unlike PIR sensors).

Would you like to adapt this for a bedroom, or maybe integrate it with your Tapo plugs or Home Assistant scenes?