Switch Component

The `switch` domain includes all platforms that should show up like a switch and can only be turned ON or OFF.

Base Switch Configuration

```yaml
switch:
  - platform: ...
    name: "Switch Name"
    icon: "mdi:restart"
```

Configuration variables:

- **name** (**Required**, string): The name of the switch.
- **icon** (*Optional*, icon): Manually set the icon to use for the sensor in the frontend.
- **inverted** (*Optional*, boolean): Whether to invert the binary state, i.e. report ON states as OFF and vice versa. Defaults to `false`.
- **internal** (*Optional*, boolean): Mark this component as internal. Internal components will not be exposed to the frontend (like Home Assistant). Only specifying an `id` without a `name` will implicitly set this to true.
- **restore_mode** (*Optional*): Control how the switch attempts to restore state on bootup. **NOTE** : Not all components consider **restore_mode**. Check the documentation of the specific component to understand how this feature works for a particular component or device. For restoring on ESP8266s, also see `restore_from_flash` in the esp8266 section.
    - `RESTORE_DEFAULT_OFF` - Attempt to restore state and default to OFF if not possible to restore.
    - `RESTORE_DEFAULT_ON` - Attempt to restore state and default to ON.
    - `RESTORE_INVERTED_DEFAULT_OFF` - Attempt to restore state inverted from the previous state and default to OFF.
    - `RESTORE_INVERTED_DEFAULT_ON` - Attempt to restore state inverted from the previous state and default to ON.
    - `ALWAYS_OFF` - Always initialize the switch as OFF on bootup.
    - `ALWAYS_ON` - Always initialize the switch as ON on bootup.
    - `DISABLED` - Does nothing and leaves it up to the downstream platform component to decide. For example, the component could read hardware and determine the state, or have a specific configuration option to regulate initial state.

Unless a specific platform defines another default value, the default is `RESTORE_DEFAULT_OFF`.

- **on_turn_on** (*Optional*, Action): An automation to perform when the switch is turned on. See switch.on_turn_on / switch.on_turn_off Trigger.
- **on_turn_off** (*Optional*, Action): An automation to perform when the switch is turned off. See switch.on_turn_on / switch.on_turn_off Trigger.
- **disabled_by_default** (*Optional*, boolean): If true, then this entity should not be added to any client's frontend, (usually Home Assistant) without the user manually enabling it (via the Home Assistant UI). Requires Home Assistant 2021.9 or newer. Defaults to `false`.
- **entity_category** (*Optional*, string): The category of the entity. See https://developers.home-assistant.io/docs/core/entity/#generic-properties for a list of available options. Requires Home Assistant 2021.11 or newer. Set to `""` to remove the default entity category.
- **device_class** (*Optional*, string): The device class for the switch. See https://developers.home-assistant.io/docs/core/entity/switch/#available-device-classes for a list of available options. Requires Home Assistant 2022.3 or newer.
- If MQTT enabled, All other options from MQTT Component.

`switch.toggle` Action

This action toggles a switch with the given ID when executed.

```
on_...:
  then:
    - switch.toggle: relay_1
```

`switch.turn_on` Action

This action turns a switch with the given ID on when executed.

```
on_...:
  then:
    - switch.turn_on: relay_1
```

`switch.turn_off` Action

This action turns a switch with the given ID off when executed.

```
on_...:
  then:
    - switch.turn_off: relay_1
```

`switch.is_on` / `switch.is_off` Condition

This Condition checks if the given switch is ON (or OFF).

```
# In some trigger:
on_...:
  if:
    condition:
      # Same syntax for is_off
      switch.is_on: my_switch
```

lambda calls

From lambdas, you can call several methods on all switches to do some advanced stuff (see the full API Reference for more info).

- `publish_state()`: Manually cause the switch to publish a new state and store it internally. If it's different from the last internal state, it's additionally published to the frontend.

  ```
  // Within lambda, make the switch report a specific state
  id(my_switch).publish_state(false);
  id(my_switch).publish_state(true);
  ```

- `state`: Retrieve the current state of the switch.

  ```
  // Within lambda, get the switch state and conditionally do something
  if (id(my_switch).state) {
    // Switch is ON, do something here
  } else {
    // Switch is OFF, do something else here
  }
  ```

- `turn_off()`/`turn_on()`: Manually turn the switch ON/OFF from code. Similar to the `switch.turn_on` and `switch.turn_off` actions, but can be used in complex lambda expressions.

  ```
  id(my_switch).turn_off();
  id(my_switch).turn_on();
  // Toggle the switch
  id(my_switch).toggle();
  ```

`switch.on_turn_on` / `switch.on_turn_off` Trigger

This trigger is activated each time the switch is turned on. It becomes active right after the switch component has acknowledged the state (e.g. after it switched ON/OFF itself).

```yaml
switch:
  - platform: gpio  # or any other platform
    # ...
    on_turn_on:
    - logger.log: "Switch Turned On!"
    on_turn_off:
    - logger.log: "Switch Turned Off!"
```

See Also

- API Reference
- Edit this page on GitHub

- BLE Client Switch
- Custom Switch
- Factory Reset Switch
- GPIO Switch
- Modbus Controller Switch
- Nextion Switch Component
- Generic Output Switch
- Restart Switch
- Safe Mode Switch
- Shutdown Switch
- Template Switch
- Tuya Switch
- UART Switch