ESPHome

Overview

> DISCLAIMER: This software is neither affiliated with nor endorsed by either Control4 or ESPHome.

Integrate ESPHome-based devices into Control4. ESPHome is an open-source system that transforms common microcontrollers, like ESP8266 and ESP32, into smart home devices through simple YAML configuration. ESPHome devices can be set up, monitored, and controlled using a web browser, Home Assistant, or other compatible platforms. This driver monitors and controls ESPHome devices directly from your Control4 system.

Index

- System Requirements
- Features
- Compatibility
  - Verified Devices
  - Supported ESPHome Entities
- Installer Setup
  - DriverCentral Cloud Setup
  - Driver Installation
  - Driver Setup
    - Driver Properties
      - Cloud Settings
      - Driver Settings
      - Device Settings
      - Bluetooth Proxy Settings
      - Button LEDs
      - Device Info
    - Driver Actions
  - Programming Reference
- Configuration Guides
  - Garage Door Configuration Guide

    - Bluetooth Proxy Configuration Guide
    - Button LED Configuration Guide
- Developer Information
- Support
- Changelog

System Requirements

- Control4 OS 3.3+

Features

- Local network communication requiring no cloud services
- Real-time updates from all supported entities exposed by the device
- Supports encrypted connections using the device encryption key
- Variable Programming Support

Compatibility

Verified Devices

This driver will generically work with any ESPHome device, but we have tested extensively with the following devices:

- ratgdo
- konnected Smart Garage Door Opener - blaQ / White

Both are relay-based garage door controllers - see the Garage Door Configuration Guide.

If you try this driver on a product listed above, and it works, let us know!

Supported Bluetooth Devices

When used as a Bluetooth proxy, this driver supports the following BLE device types through sub-drivers:

Protocol	Sub-Driver	Example Devices
SwitchBot	ESPHome SwitchBot	Bot, Plug Mini, Relay Switch, Meter, Motion, Contact
BTHome	ESPHome BTHome	Shelly BLU Button/Door/Motion/H&T, DIY sensors
Govee	ESPHome Govee	Temperature/humidity monitors, meat thermometers
Yale/August	ESPHome Yale	Yale and August smart locks
SereneScent	ESPHome SereneScent	Homedics SereneScent Diffuser

See the individual sub-driver documentation for device-specific details.

Supported ESPHome Entities

Entity Type	Supported
Alarm Control Panel	✗
API Noise	✓
Binary Sensor	✓
Bluetooth Proxy	✓
Button	✓
Camera	✗
Climate	✓
Cover	✓
Date	✓
Datetime	✓
Event	✓
Fan	✓
Light	✓
Lock	✓

Entity Type	Supported
Media Player	✗
Number	✓
Select	✓
Sensor	✓
Siren	✗
Switch	✓
Text	✓
Text Sensor	✓
Time	✓
Update	✗
Valve	✗
Voice Assistant	✗*
Water Heater	✓

> * Voice Assistant requires a speech-to-text and intent processing pipeline (e.g. Home Assistant Assist). Control4 does not natively provide voice intent handling, so this entity type is not supported.

Installer Setup

> ⚠ Only a ***single*** driver instance is required per ESPHome device. Multiple instance of this driver connected to the same device will have unexpected behavior. However, you can have multiple instances of this driver connected to ***different*** ESPHome devices.

DriverCentral Cloud Setup

> If you already have the DriverCentral Cloud driver installed in your project you can continue to Driver Installation.

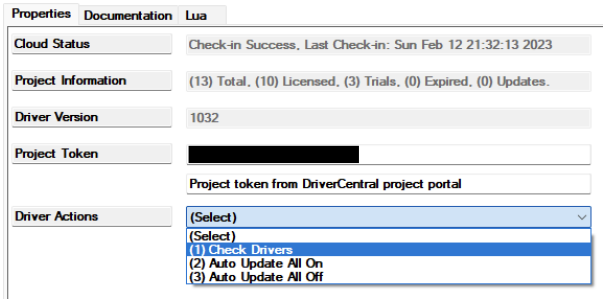
This driver relies on the DriverCentral Cloud driver to manage licensing and automatic updates. If you are new to using DriverCentral you can refer to their Cloud Driver documentation for setting it up.

Driver Installation

1. Download the latest `control4-esphome.zip` from DriverCentral.
2. Extract and install all `.c4z` files.
3. Use the "Search" tab to find the "ESPHome" driver and add it to your project.

   > ⚠ A ***single*** driver instance is required per ESPHome device.

4. Select the newly added driver in the "System Design" tab. You will notice that the `Cloud Status` reflects the license state. If you have purchased a license it will show `License Activated`, otherwise `Trial Running` and remaining trial duration.
5. You can refresh license status by selecting the "DriverCentral Cloud" driver in the "System Design" tab and perform the "Check Drivers" action.



6. Configure the Device Settings with the connection information.
7. After a few moments the `Driver Status` will display `Connected`. If the driver fails to connect, set the `Log Mode` property to `Print` and re-set the `IP Address` field to reconnect. Then check the lua output window for more information.
8. Once connected, the driver will automatically create variables and connection bindings for each supported entity type.
9. To control climate, lights, fans, locks, and/or water heaters, use the "Search" tab to find the "ESPHome Climate", "ESPHome Light", "ESPHome Fan", and/or "ESPHome Lock" driver. For fans, choose the speed variant that matches your fan (e.g., "ESPHome Fan (3 Speed)"). Water heater entities use the "ESPHome Climate" driver. Add one driver instance for each exposed entity in your project. In the "Connections" tab, select the "ESPHome" driver and bind the entities to the newly added drivers.

Driver Setup

Driver Properties

Cloud Settings

Cloud Status (read-only)

Displays the DriverCentral cloud license status. Without an active license, Driver Status reads `No active license`: the driver disconnects from the device and shows its lights, fans, locks and thermostats offline, a light at the level it had, until a license is active again. The driver stops once five minutes pass without a license, and connects again as soon as one is active.

Automatic Updates [Off | *On*]

Enables or disables automatic driver updates via DriverCentral. Changing it on any ESPHome driver or sub-driver changes it on all of them.

Driver Settings

Driver Status (read-only)

Displays the current status of the driver.

Driver Version (read-only)

Displays the current version of the driver.

Log Level [0 - Fatal | 1 - Error | 2 - Warning | *3 - Info* | 4 - Debug | 5 - Trace | 6 - Ultra]

Sets the logging level. Default is `3 - Info`.

Log Mode [*Off* | Print | Log | Print and Log]

Sets the logging mode. Default is `Off`.

Device Log Forwarding [*Off* | On]

Forward ESPHome device logs to the driver's Lua output at the current Log Level. Changing Log Level or disabling Log Mode will reconnect to apply the new settings; a light keeps its level while it does.

Support Code

Leave it blank unless Finite Labs support gives you a code: it opens Composer's Lua tab to this driver for an hour.

Cloud Monitoring [Disabled | *Enabled*]

When Enabled, the driver sends Finite Labs a short report once an hour, and a report of each error it hits, at most 10 a minute. We use them to find and fix problems in the driver. Set it to Disabled to send nothing.

A report holds the controller's MAC address and OS version, the driver's name, version, device ID and DriverCentral license status, how much memory the driver uses, and the sizes of its largest internal tables. An error report also holds the error message and where in the driver it happened. Before a message is sent, anything in it that looks like a password, token, key, network address or email address is removed.

Device Settings

IP Address

Sets the device IP address (e.g. `192.168.1.30`). Domain names are allowed as long as they can be resolved to an accessible IP address by the controller. HTTPS is not supported.

> ⚠ If you are using an IP address, you should ensure it will not change by assigning a static IP or creating a DHCP reservation.

Port [1 - 65535, default: *6053*]

Sets the device port. The default port for ESPHome devices is `6053`.

Authentication Mode [*None* | Password | Encryption Key]

Selects the authentication method for connecting to the ESPHome device.

- **None**: No authentication required.
- **Password**: Use a password for authentication (see below).
- **Encryption Key**: Use an encryption key for secure communication (see below).

> **Tip:** For ESPHome devices used primarily as Bluetooth proxies, consider configuring the firmware without API encryption. In busy BLE environments, the volume of proxy traffic combined with the controller's limited cryptographic performance can cause significant CPU load. BLE traffic is inherently over-the-air, and sensitive protocols (such as Yale/August lock commands) use their own end-to-end encryption between

> the driver and the device, making it safe to relay through an unencrypted proxy. To disable encryption, omit the `encryption` block from the ESPHome `api:` configuration and set Authentication Mode to `None`.

Password

Shown only if Authentication Mode is set to `Password`. Sets the device password. This must match the password configured on the ESPHome device.

Encryption Key

Shown only if Authentication Mode is set to `Encryption Key`. Sets the device encryption key for secure communication. This must match the encryption key configured on the ESPHome device.

Use OpenSSL [*Yes* | No]

Use OpenSSL for encryption. This should typically be left at the default value of `Yes` for better performance and compatibility.

Bluetooth Proxy Settings

> The Bluetooth Proxy feature requires an ESP32 device with the `bluetooth_proxy` component configured in ESPHome. See the ESPHome Bluetooth Proxy documentation for firmware configuration.

Bluetooth Proxy Status (read-only)

Shows the current state of the Bluetooth proxy. The format is a pipe-separated list of status components:

Standalone Mode: `Standalone Mode | Scanning (Passive) | 1/4 Active`

Coordinator Mode: `Coordinator Mode | Scanning (Passive) | 0/3 Active | MAC Filter: 5 device(s)`

Components explained:

- **Mode** - "Standalone Mode" or "Coordinator Mode" depending on whether the driver is connected to a Bluetooth Coordinator
- **Scanner State** - Current state (Scanner Idle, Starting Scan, Scanning, Stopping Scan, Scanner Stopped or Scan Failed) and mode (Passive or Active)

- **Connection Slots** - Shows "used/total Active" slots (e.g., "1/4 Active" means 1 slot in use out of 4 available). Passive devices use no slot, and a device that connects only briefly frees its slot after each exchange. If you select more active devices than available slots, "(Oversubscribed)" is appended (see Oversubscription)
- **MAC Filter** - (Coordinator mode only) Shows how many device MACs are being filtered for, or "none" if forwarding all advertisements
- **Not Recovering - Power Cycle Device** - The proxy is wedged and only a restart of the ESP32 will clear it: the scanner has stopped receiving advertisements and the driver cannot restart it, or a device the proxy still holds connected has stopped answering. Bound devices report "Bluetooth proxy needs a restart". Press the device's Restart button or power cycle it, or let Restart Bluetooth Proxy When Stuck do it

Bluetooth Proxy Capabilities (read-only)

Displays a comma-separated list of capabilities supported by this ESPHome Bluetooth proxy:

- **Scan** - Can receive BLE advertisements (passive scanning)
- **Connect** - Can establish GATT connections to devices (active connections)
- **Cache** - Caches GATT service data remotely
- **Pair** - Can pair with devices requiring authentication
- **Raw** - Can receive raw advertisement data

Select Bluetooth Devices

> Only visible in **Standalone Mode** (hidden when connected to a Bluetooth Coordinator, as device selection is done there instead).

A dropdown list showing discovered BLE devices. Select "Refresh List" to start a new scan.

During scanning:

- The dropdown displays "-- Scanning..." while the scan is in progress
- Select "-- Stop Scan" to stop early and keep newly discovered devices
- Select "-- Abort Scan" to stop early and discard newly discovered devices
- When complete, the dropdown repopulates with discovered devices

Device list shows:

- MAC Address
- Device Name (if available)
- Device Type (BTHome, SwitchBot, Govee, etc.)

• Connection Type (Active/Passive)

After selecting a device, a connection binding is automatically created for the appropriate sub-driver.

> **Tip:** Some BLE devices (buttons, sensors with long advertisement intervals) may be in sleep mode. Wake them by pressing buttons, triggering motion sensors, or opening/closing contact sensors during the scan to ensure they appear in the device list.

Bluetooth Scan Duration [5 - 60, default: *30*]

> Only visible in **Standalone Mode**.

Sets how long (in seconds) to scan for BLE devices when refreshing the device list. Longer scans may discover more devices with long advertisement intervals.

Bluetooth Proxy Room

> Only visible in **Coordinator Mode** (when connected to a Bluetooth Coordinator).

Sets the room where this Bluetooth proxy is physically located. This is used for presence tracking to determine which room a device is in based on signal strength.

Minimum Room RSSI Override (dBm) [-100 - -40, default: *-100*]

> Only visible in **Coordinator Mode** (when connected to a Bluetooth Coordinator).

Overrides the coordinator's global "Minimum Room RSSI" setting for this proxy's room. Use this when a room has different size or characteristics than others.

- **-100 (default)** - Use the coordinator's global setting
- **-85** - More permissive; allow detection from further away (large rooms)
- **-60** - More restrictive; require closer proximity (small rooms)

Examples:

- Large living room: Set to `-85` to allow detection from further away
- Small bathroom: Set to `-60` to require closer proximity
- Leave at `-100` to use the coordinator's global setting

> **Note:** Only affects room assignment for this proxy. The value is sent to the Bluetooth Coordinator when this proxy connects or when the setting changes.

Restart Bluetooth Proxy When Stuck [default: *Off*]

> Only visible when the ESPHome device has a `restart` button with its default icon.

The restart button to press when the proxy is wedged (see Bluetooth Proxy Status): its scanner has been stuck for about 15 minutes, or a device it holds connected has failed service discovery three times in a row. The button is pressed at most once every 6 hours.

- **Off (default)** - Only report the condition in Bluetooth Proxy Status
- ***button name*** - Press this button

Only buttons that report the `restart` device class, the config entity category and the `mdi:restart` icon are offered, so ESPHome's Safe Mode and Factory Reset buttons never are. A Restart button whose icon the YAML overrides is not offered either.

Button LEDs

> Only visible when the device has both event entities and light entities.

An event bound to a load acts as a keypad button. These properties choose which of the device's lights is that button's LED, which then shows the load the way a Control4 keypad lights its buttons. See the Button LED Configuration Guide.

Configure Button LED

Lists each event entity, with its LED when one is chosen (e.g., `Gang 3 Paddle Event -> Gang 3 LED`). Select one to choose its LED below.

Button LED

The light that is the selected button's LED, or **(None)** to leave the light alone. A button and light that are alone together on an ESPHome sub-device are paired automatically.

Device Info

Name (read-only)

Displays the name of the connected ESPHome device.

Model (read-only)

Displays the model of the connected ESPHome device.

Manufacturer (read-only)

Displays the manufacturer of the connected ESPHome device.

MAC Address (read-only)

Displays the MAC address of the connected ESPHome device.

Firmware Version (read-only)

Displays the firmware version of the connected ESPHome device.

Driver Actions

Reset Driver

> ⚠ This will remove all connection bindings and delete the variables and events, along with any programming on them.

Resets the driver to its initial state, removing all dynamically created connections, variables and events. It forgets each button's LED light and the Bluetooth proxy's restart button, and returns the settings to their defaults. The device's address and login stay. This is useful if you change the connected ESPHome device or there are stale connections or variables.

It also turns Cloud Monitoring back on, as a fresh install has it.

Parameters:

- **Are You Sure?** [***No*** | Yes] - Confirmation to reset the driver.

Diagnostics Snapshot

Prints a report in the Lua output, and sends it to Finite Labs only while Cloud Monitoring is Enabled.

Programming Reference

Once connected, the driver automatically creates variables and bindings for each supported ESPHome entity. Use this reference for Control4 programming.

Driver Variables

In addition to per-entity variables, the driver publishes the following device-level variables. `Connected` tracks the live connection; the rest mirror the matching read-only Device Info properties.

Variable	Type	Description
Connected	BOOL	True while the driver is talking to the device
Name	STRING	Friendly name reported by the ESPHome device
Model	STRING	Device model string reported by ESPHome
Manufacturer	STRING	Manufacturer string reported by ESPHome
MAC Address	STRING	ESPHome device MAC address

> **Tip:** Every sub-driver publishes its own `Connected` variable too, tracking the entity it is bound to. Use it in Programming to drive a custom Navigator element, send a notification, or trigger anything else when a device drops off. Thermostats, lights and fans additionally report their connection state to their proxy, so Navigator greys them out on its own.

Variables by Entity Type

Entity Type	Variable Name	Type	Notes
Binary Sensor	`{name} State`	BOOL	"1" = triggered, "0" = clear
Button	(none)	-	Use "Press Button" command (see below)
Climate	(none)	-	State via Thermostat proxy
Cover	`{name} State`	STRING	"open", "closed", "opening", "closing"
Date	`{name}`	STRING	Writable, formatted as YYYY-MM-DD
Datetime	`{name}`	STRING	Writable, formatted as YYYY-MM-DD HH:MM:SS
Event	`{name} Last Event`	STRING	Last event type (e.g., "single_press")
Event	`{name} LED`	BOOL	"1" = the button's linked load is lit
Fan	(none)	-	State via Fan proxy
Light	(none)	-	State via Light proxy
Lock	(none)	-	State via Lock proxy

Entity Type	Variable Name	Type	Notes
Number	`{name}`	NUMBER	Writable, 1 decimal precision
Select	`{name}`	STRING	Writable, current option
Sensor	`{name}`	NUMBER	Read-only, 1 decimal precision
Switch	`{name} State`	BOOL	"1" = on, "0" = off (writable)
Text	`{name}`	STRING	Writable
Text Sensor	`{name}`	STRING	Read-only
Time	`{name}`	STRING	Writable, formatted as HH:MM:SS
Water Heater	(none)	-	State via Thermostat proxy

> **Note:** `{name}` is the entity's ESPHome name. An entity with no name uses its device's name. Entities of different types may share a name. When two of one type share a name, or two would write the same variable, the others are told apart by their sub-device or their type (e.g., "Kitchen Temperature" or "Status (Sensor)").

A writable variable shows the device's value. A value set while the device is offline, or a date or time in another format, goes back to the device's value.

Bindings by Entity Type

Entity Type	Binding Class	Purpose
Binary Sensor	`CONTACT_SENSOR`	Integrates with Contact Sensor proxy
Switch	`RELAY`	Control via Relay proxy
Cover	`CONTACT_SENSOR`	Open/closed state contacts
Cover	`RELAY`	Open/close/stop control relays
Button	`BUTTON_LINK`	Allows other devices to trigger button
Event	`BUTTON_LINK`	One connection per event type, triggers other devices

Entity Type	Binding Class	Purpose
Climate	`ESPHOME_CLIMATE`	Bind to ESPHome Climate sub-driver **
Fan	`ESPHOME_FAN_N_SPEED[_REVERSE]`	Bind to ESPHome Fan sub-driver
Light	`ESPHOME_LIGHT`	Bind to ESPHome Light sub-driver
Lock	`ESPHOME_LOCK`	Bind to ESPHome Lock sub-driver
Water Heater	`ESPHOME_CLIMATE`	Bind to ESPHome Climate sub-driver

> **Note:** Sensor, Number, Select, Text, Text Sensor, Date, Time and Datetime entities do not create bindings. They expose data only through variables and events.

> **Note:** The Button and Event binding classes are the same but point opposite ways. A Button binding lets another device press the ESPHome button; an Event binding lets the ESPHome device act as a keypad button and drive a load.

> **Note:** Water Heater entities share the `ESPHOME_CLIMATE` binding class and are controlled via the ESPHome Climate sub-driver. Device modes (such as Eco, Heat Pump, Gas) are exposed as presets.

> ** **Climate remote sensor:** The Climate sub-driver supports using an external Control4 temperature sensor instead of the climate device's built-in sensor. This feature requires the ESPHome device YAML to define user-defined API services (e.g., `set_remote_temperature` and `use_internal_temperature`). See the ESPHome Climate sub-driver documentation for full details and YAML examples.

Bluetooth Proxy Bindings

When the connected ESPHome device exposes a Bluetooth proxy, additional dynamic bindings are created separately from the entity bindings above:

Source	Binding Class	Purpose
Bluetooth Coordinator link	`ESPHOME_BLUETOOTH`	Connects the ESPHome driver to the Bluetooth Coordinator

Source	Binding Class	Purpose
Selected BTHome device	`ESPHOME_BTHOME`	Bind to ESPHome BTHome sub-driver
Selected Govee device	`ESPHOME_GOVEE`	Bind to ESPHome Govee sub-driver
Selected SereneScent device	`ESPHOME_SERENESCENT`	Bind to ESPHome SereneScent sub-driver
Selected SwitchBot device	`ESPHOME_SWITCHBOT`	Bind to ESPHome SwitchBot sub-driver
Selected Yale/ August lock	`ESPHOME_YALE`	Bind to ESPHome Yale sub-driver

> **Note:** Per-device Bluetooth bindings are created automatically when a device is chosen via the `Select Bluetooth Devices` property (standalone mode) or via the Bluetooth Coordinator's device selector (coordinator mode). Binding display names include the device MAC address suffix for easy identification.

Events by Entity Type

Entity Type	Event Name	Description
Event	`{name}: {event_type}`	One C4 event per event type for programming

> **Note:** Event entities are stateless triggers (button presses, gestures, doorbell rings). Each discovered event type creates a Control4 event that can be used in programming, plus a `BUTTON_LINK` connection named `{name} {event_type}` that can be bound straight to a load. The `{name} Last Event` variable tracks the most recent event type, and `{name} LED` whether the button's linked load is lit (see Button LEDs).

Commands

Command	Parameters	Description
Press Button	Button	Triggers an ESPHome button entity by name

Command	Parameters	Description
Set Select	Select, Option	Sets a select entity to the specified option

> **Note:** The Button parameter is a dynamic list populated with discovered ESPHome button entities. The Select and Option parameters are dynamic lists populated with discovered ESPHome select entities and their available options.

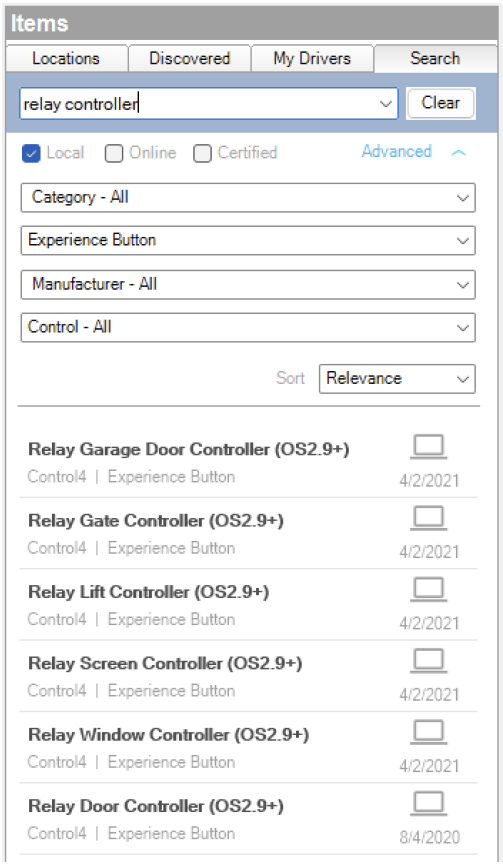
Garage Door Configuration Guide

This guide provides instructions for configuring the ESPHome driver for garage door control via relays in Control4 Composer Pro. It applies to any ESPHome device that exposes a cover entity, including ratgdo and konnected's Smart Garage Door Opener.

The driver names the relay and contact bindings after the cover entity, so the labels in Composer Pro depend on the device. ratgdo names its cover "Door", so you will see "Open Door" and "Door Closed". konnected defaults to "Garage Door". Substitute your device's cover name throughout this guide.

Add Relay Controller Driver

Add the desired relay controller driver to your Control4 project in Composer Pro.



Relay Controller Properties

The device exposes a "Cover" entity in ESPHome, which maps to the relay controller functionality in Control4.

Number of Relays

The driver creates separate relays for opening and closing the garage door. In Composer Pro, you should configure the relay settings as follows:

- Set to **2 Relays** (Open/Close) or **3 Relays** (Open/Close/Stop)
  - The stop relay is only created when the device reports support for it. If you are not sure, look at the connections in Composer Pro to see if the "Stop" relay is available, and configure for 3 relays if it is.

Relay Configuration

- Set to **Pulse**
  - These devices use momentary pulses to trigger the garage door opener, similar to a wall button press

Pulse Time

- Set all relay pulse times to **500** (default)
    - This is the duration the relay will be activated

Invert Relay

- Set all invert relay properties to **No** (default)

Contact Debounce

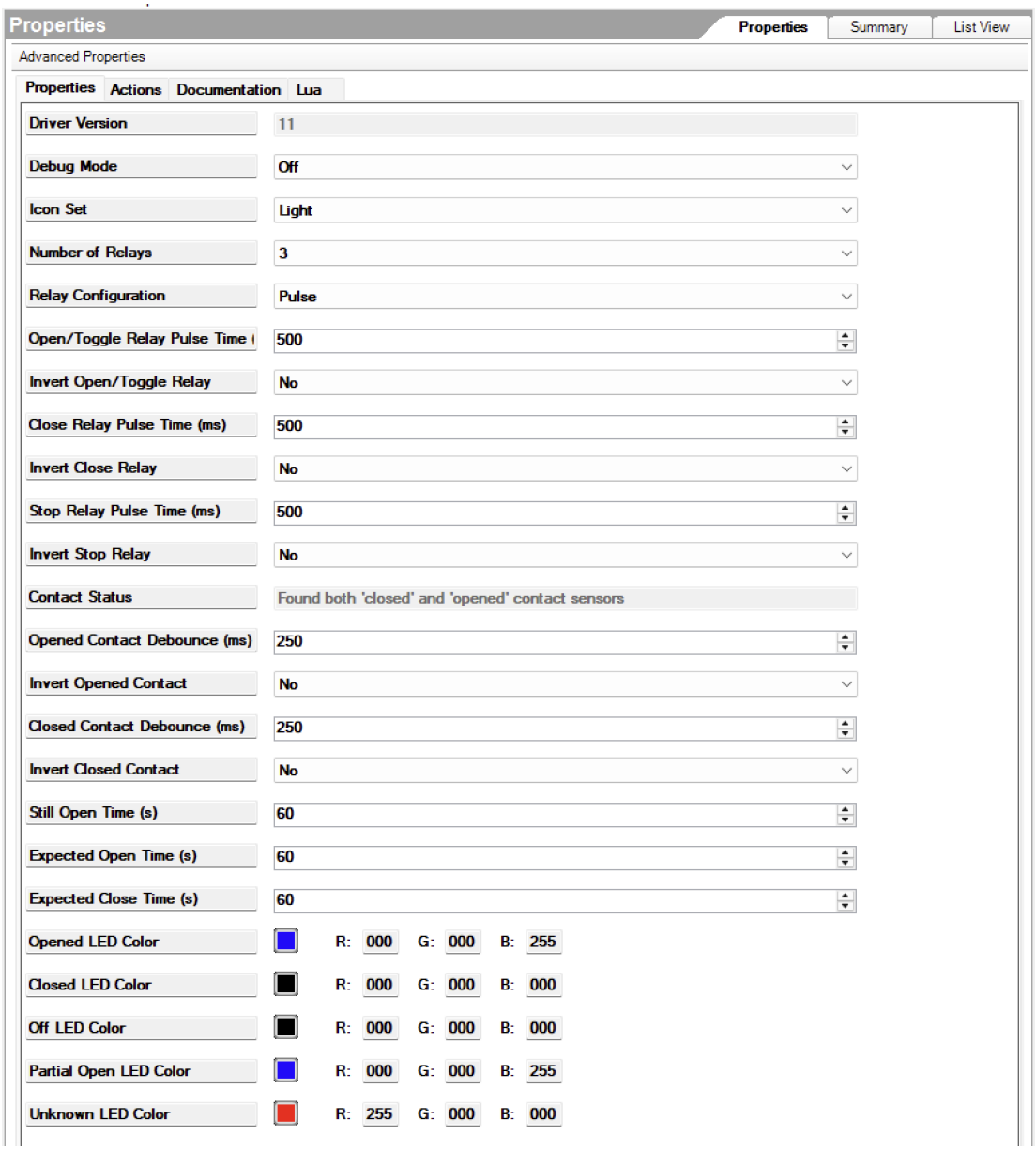
- Set all contact debounce times to **250** (default)
    - This helps prevent false flapping of the garage door state sensors

Invert Contact

- Set all invert contact properties to **No** (default)

Example Properties

For reference, here is an example of the relay controller properties in Composer Pro:



Relay Controller Connections

Relays

Where `{cover}` is the name of the device's cover entity ("Door" on ratgdo, "Garage Door" on konnected):

- **Open**: Connect to the "Open {cover}" relay
- **Close**: Connect to the "Close {cover}" relay
- **Stop**: Connect to the "Stop {cover}" relay, if available

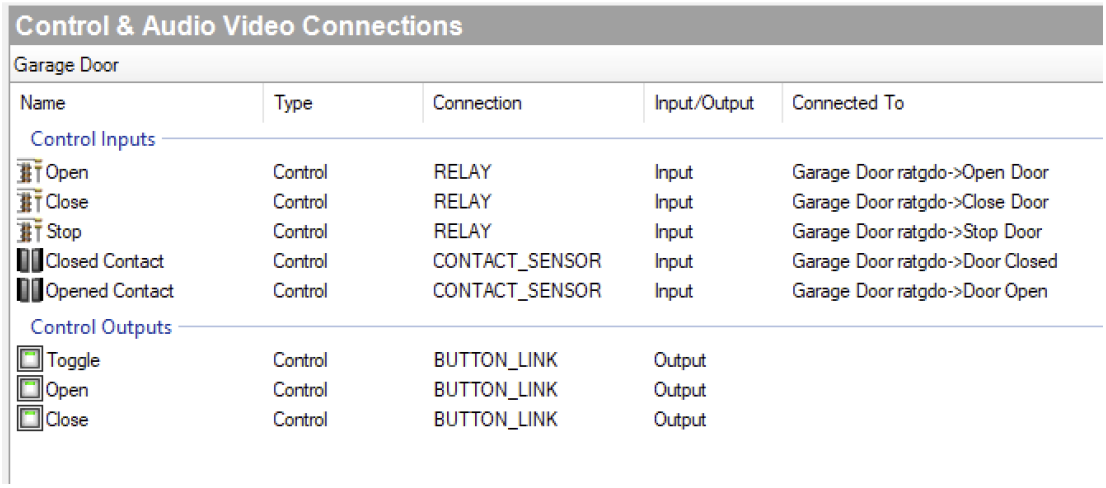
Contact Sensors

- **Closed Contact**: Connect to the "{cover} Closed" contact

- **Opened Contact**: Connect to the "{cover} Open" contact

Example Connections

For reference, here is an example of how the connections should look in Composer Pro:



Control & Audio Video Connections

Garage Door

Name	Type	Connection	Input/Output	Connected To
Control Inputs				
Open	Control	RELAY	Input	Garage Door ratgdo->Open Door
Close	Control	RELAY	Input	Garage Door ratgdo->Close Door
Stop	Control	RELAY	Input	Garage Door ratgdo->Stop Door
Closed Contact	Control	CONTACT_SENSOR	Input	Garage Door ratgdo->Door Closed
Opened Contact	Control	CONTACT_SENSOR	Input	Garage Door ratgdo->Door Open
Control Outputs				
Toggle	Control	BUTTON_LINK	Output	
Open	Control	BUTTON_LINK	Output	
Close	Control	BUTTON_LINK	Output	

Programming

You can create programming in Control4 to:

- Open/close the garage door based on events
- Monitor the garage door state
- Set up notifications for garage door status changes
- Create custom buttons on touchscreens and remotes

Example: Creating a "Still Open" Alert

Using the "Still Open Time" property from the relay controller driver:

1. Set the "Still Open Time" to your desired duration (e.g., 10 minutes)
2. Create a programming rule that triggers when the "Still Open" event fires
3. Add actions to send notifications or perform other tasks

Additional Entities

Depending on your device, firmware, and its capabilities, there may be additional entities exposed by the ESPHome driver. These can come as additional connections or driver variables.

Please refer to your device's documentation for more information on specific entities:

- ratgdo: https://ratgdo.github.io/esphome-ratgdo/webui_documentation.html
- konnected: https://support.konnected.io/esphome-on-the-konnected-garage-door-opener

Bluetooth Proxy Configuration Guide

This guide explains how to use the ESPHome Bluetooth Proxy feature to integrate BLE devices into Control4.

Prerequisites

- ESP32 device with ESPHome firmware and `bluetooth_proxy` component enabled
- BLE devices within range of the ESP32

Recommended Hardware:

The following POE-powered Bluetooth proxies are excellent choices with 4 active connection slots:

- Seeed Studio XIAO W5500 Ethernet Adapter
- Olimex ESP32-POE or ESP32-POE-ISO

Some hardware links above are affiliate links. Purchases through them help support driver development at no extra cost to you.

Firmware Installation:

- **Quick Start:** Use web.esphome.io for one-click firmware installation directly in your browser - no YAML configuration needed.
- **Advanced:** For more control over settings, create a custom YAML configuration. See the ESPHome Bluetooth Proxy documentation for configuration options.

Understanding Connection Types

BLE devices use one of two connection modes:

Passive Mode (No Slot Required)

Passive devices broadcast their data in advertisements. The proxy listens without establishing a connection. These devices include:

- **Shelly BLU devices** - Button, Door/Window, Motion, H&T sensors (native BTHome support)
- **BTHome sensors** - Temperature, humidity, motion, door sensors
- **Govee sensors** - Temperature/humidity monitors, meat thermometers
- **SwitchBot sensors** - Meters, motion sensors, contact sensors, water leak

Advantage: Unlimited passive devices can be monitored simultaneously.

Active Mode (Uses Connection Slot)

Active devices require a GATT connection to send commands. The ESP32 has limited connection slots (typically 3-4). These devices include:

- **SwitchBot Bot** - Requires connection to send press/on/off commands
- **SwitchBot Switch** - Plug Mini, Relay switches (encrypted commands)
- **Yale/August Locks** - Requires connection for encrypted lock/unlock commands
- **Homedics SereneScent** - Requires connection for commands and status reads

Oversubscription

You can select more active devices than available connection slots. This is called **oversubscription** and works well when devices only need brief connections to exchange data (e.g., sending a command to a SwitchBot Bot). The device connects, sends the command, and immediately frees the slot.

Oversubscription becomes problematic when multiple devices need simultaneous connections. If all slots are in use, commands to additional devices will queue and retry until a slot becomes available.

> **Tip:** If you see "Allocation failed" errors in the logs, you have too many concurrent active connections. Consider reducing the number of active devices or adding another proxy.

Step-by-Step Setup

1. **Add the ESPHome driver** and configure it to connect to your ESP32 device.

2. **Verify Bluetooth Proxy Status** shows "Scanning" with free slots.
3. **Scan for devices:**
    - Set "Bluetooth Scan Duration" (30 seconds recommended)
    - Select "Refresh List" from the "Select Bluetooth Devices" dropdown
    - Wait for the scan to complete
4. **Select a discovered device** from the dropdown. A connection will be automatically created.
5. **Add the appropriate sub-driver:**
    - Search for the driver matching your device type (e.g., "ESPHome BTHome")
    - Add it to your project
6. **Bind the sub-driver** to the connection created in step 4.
7. **Configure the sub-driver** properties as needed.

Supported Device Types

Device Protocol	Sub-Driver	Connection
BTHome	ESPHome BTHome	Passive
Govee	ESPHome Govee	Passive
SwitchBot	ESPHome SwitchBot	Active/Passive
Yale/August	ESPHome Yale	Active
SereneScent	ESPHome SereneScent	Active

Performance Considerations

Device Limits

The ESP32 Bluetooth proxy has practical limits on device capacity:

Connection Type	Recommended Limit	Notes
Passive devices	20-30	Sensors broadcasting advertisements

Connection Type	Recommended Limit	Notes
Active connections	3-5	Devices requiring GATT connections

Exceeding these limits may cause:

- Missed advertisements from passive devices
- "Allocation failed" errors for active connections
- Delayed or failed commands to active devices

> **Warning:** If you see "Too many BLE events to process" in the logs, the proxy is overwhelmed. Reduce the number of tracked devices or add another proxy.

Busy BLE Environments

In environments with many BLE devices (smart home hubs, fitness equipment, wireless speakers, neighbors' devices), performance may degrade:

- **2.4 GHz congestion** - BLE and WiFi share spectrum; heavy WiFi traffic affects BLE reception
- **Advertisement flooding** - Many devices broadcasting simultaneously can overwhelm the scanner
- **Interference sources** - Microwaves, USB 3.0 devices, and poorly shielded electronics cause interference

Mitigation strategies:

1. Use Ethernet-connected ESP32 boards (Olimex ESP32-POE) to avoid WiFi/BLE contention
2. Position proxies away from WiFi routers (at least 2-3 meters)
3. Use the Bluetooth Coordinator to distribute load across multiple proxies
4. Reduce scan duration if not actively discovering new devices

Proxy Placement Tips

For optimal BLE reception:

- **Height matters** - Place at chest height or higher, not on the floor
- **Avoid metal** - Keep away from refrigerators, filing cabinets, and metal shelving (metal causes reflections and unstable signals)
- **Avoid enclosed spaces** - Don't place inside cabinets or behind furniture

- **Distance from electronics** - Keep 2-3 meters from routers, switches, and other network equipment

Coverage expectations:

Environment	Typical Range
Open indoor space	10-15 meters (30-50 ft)
Through 1-2 walls	5-10 meters (15-30 ft)
Concrete/brick walls	3-5 meters (10-15 ft)

Bluetooth Coordinator Setup

For advanced setups with **multiple ESPHome Bluetooth proxies**, use the **ESPHome Bluetooth Coordinator** driver to aggregate them. The coordinator provides:

- **RSSI-based routing** - Commands are routed to the proxy with the best signal strength for each BLE device
- **Automatic failover** - If a connection fails, the coordinator retries through alternate proxies
- **Room-level presence tracking** - Track which room a BLE device (like a phone) is in based on signal strength from each proxy

> **Note:** Apple devices (iPhone, iPad, Apple Watch) and Android devices with MAC randomization enabled cannot be tracked for presence. Use dedicated BLE beacons or devices with static MAC addresses instead.

When to Use the Coordinator

Setup	Recommendation
Single ESP32 proxy	Use ESPHome driver directly
Multiple proxies	Use Bluetooth Coordinator
Want presence tracking	Use Bluetooth Coordinator
Want failover/redundancy	Use Bluetooth Coordinator

Coordinator Setup Steps

1. **Add the Bluetooth Coordinator driver** to your project

2. **Connect your ESPHome drivers** to the coordinator's proxy bindings (Connections tab)
3. **Set the "Bluetooth Proxy Room" property** on each ESPHome driver to indicate where that proxy is physically located
4. **Select devices** via the coordinator's "Select Bluetooth Devices" property
5. **For presence tracking**, select devices via "Select Presence Devices"

When an ESPHome driver is connected to the coordinator:

- The "Select Bluetooth Devices" property is hidden (selection is done in the coordinator)
- The "Bluetooth Proxy Room" property becomes visible for presence tracking configuration

See the **ESPHome Bluetooth Coordinator** driver documentation for full details on presence tracking settings and events.

Button LED Configuration Guide

An ESPHome event bound to a load acts as a Control4 keypad button: pressing it drives the load. With a Button LED, the device also shows the load's state on one of its own lights, as a keypad lights its button.

Requirements

The device needs:

- **The button as an `event` entity.** Only events create button connections. A button exposed only as a `binary_sensor` appears as a contact sensor instead.
- **The LED as a `light` entity.** Any light platform works: `binary` or `monochromatic` lights turn on and off, and `rgb` (or any light that can show a color) shows the load's colors. The `status_led:` component is not a light entity; use `light: - platform: status_led` instead.

No other YAML is needed.

```yaml
binary_sensor:
  - platform: gpio
    pin: GPIO0
    id: button_1
    internal: true
    on_multi_click:
      - timing: [ON for at most 1s, OFF for at least 0.5s]
        then:
          - event.trigger: { id: button_1_event, event_type: press }
      - timing: [ON for at least 1s]
        then:
          - event.trigger: { id: button_1_event, event_type: long_press }

event:
  - platform: template
    id: button_1_event
    name: "Button 1"
    event_types: [press, long_press]

output:
  - platform: gpio
    pin: GPIO2
    id: led_1_out

light:
  - platform: binary
    name: "Button 1 LED"
    output: led_1_out
```

To pair them without any setup in Composer, put the button and its LED alone together on an ESPHome sub-device (ESPHome 2025.7 or later):

```yaml
esphome:
  devices:
    - id: button_1
      name: "Button 1"
event:
  - platform: template
    id: button_1_event
    name: "Button"
    device_id: button_1
    event_types: [press, long_press]
light:
  - platform: binary
    name: "LED"
    device_id: button_1
    output: led_1_out
```

On a display, a template output can recolor a widget instead of driving a pin:

```yaml
output:
  - platform: template
    id: button_2_led_out
    type: binary
    write_action:
      - lvgl.widget.update:
          id: lv_button_2
          bg_color: !lambda 'return lv_color_hex(state ? 0x3A3420 :
0x20252B);'
light:
  - platform: binary
    name: "Button 2 LED"
    output: button_2_led_out
```

Setup

1. Bind the event's `{name} {event_type}` connection to the load, as you would a keypad button. Bind it to a light's Toggle, On or Off button link to have the button toggle, turn on or turn off the light.
2. In the ESPHome driver's properties, select the button under **Configure Button LED** and its light under **Button LED**. A pair already made from a sub-device shows next to the button.

How the LED Behaves

- A light that can show a color shows the load's on color while the load is on and its off color while it is off; black is dark. Set the colors on the load, as for any keypad button linked to it: for a light, under its **Button and Button Connection Status LED Colors**. A load that sends no colors lights the LED blue. A light that cannot show a color is on while the load is on and off while it is off.
- When the link is bound, when the device connects and after the driver reloads, the driver asks the load for its state, so the LED is right without waiting for the load to change.
- A button has one LED but a connection for each event type. The LED follows the first event type with a load bound, in the order the device declares them (normally press). Unbinding it hands the LED to the next one; with none left it goes dark.

> **Note:** Leave a light used as a button LED unbound from an ESPHome Light driver. Both would command it, and the button's load would win each time it changes.

Developer Information



Copyright © 2026 Finite Labs LLC

All information contained herein is, and remains the property of Finite Labs LLC and its suppliers, if any. The intellectual and technical concepts contained herein are proprietary to Finite Labs LLC and its suppliers and may be covered by U.S. and Foreign Patents, patents in process, and are protected by trade secret or copyright law. Dissemination of this information or reproduction of this material is strictly forbidden unless prior written permission is obtained from Finite Labs LLC. For the latest information, please visit https://drivercentral.io/platforms/control4-drivers/utility/esphome

Support

If you have any questions or issues integrating this driver with Control4 or ESPHome, you can contact us at driver-support@finitelabs.com or call/text us at +1 (949) 371-5805.

Changelog

v20261106 - 2026-11-06

Added

- Added a `Restart Bluetooth Proxy When Stuck` property, which presses the ESPHome restart button you choose when a Bluetooth proxy stops answering, at most once every 6 hours. Off by default
- Added a `Support Code` property to the ESPHome driver and its sub-drivers. A code from Finite Labs support opens the driver's Lua tab in Composer for an hour; leave it blank otherwise
- Added a `Cloud Monitoring` property, Enabled by default, and a `Diagnostics Snapshot` action to every driver. Cloud Monitoring sends Finite Labs an hourly report of the driver's memory use and a report of each error it hits; the documentation says what they hold

Fixed

- Fixed an ESPHome climate device that does not report what it is doing never setting the thermostat's HVAC state; the state now follows the mode (Cool reads as Cooling, Heat/Cool and Auto as Idle). A water heater now reports an HVAC state too: Heating while it is on, Off while it is off
- Fixed a Bluetooth proxy that still held a device connected but no longer answered it going unreported: `Bluetooth Proxy Status` now asks for a power cycle, and the device shows "Bluetooth proxy needs a restart" instead of "GATT discovery failed"
- Fixed a SwitchBot command sent while the device was still connecting failing, and a two-channel SwitchBot relay turning on only one channel when a scene set both
- Fixed a Yale lock or unlock in Persistent mode being dropped when the connection failed; it is now retried, as it is after a dropped connection
- Fixed relay drivers bound to ESPHome switches and covers: a switch no longer turns off when a relay driver bound to it asks for its state, and a relay driver no longer runs its programming each time the device reconnects or a cover updates
- Fixed a pulsed ESPHome switch staying on when another switch on the same device was pulsed before it finished
- Fixed a light's color in Navigator and in its variables staying on the old color after a color preset was picked, or after the light turned on with Color On Mode set to Previous or to a preset; the light itself changed color

- Fixed a light reporting its capability and color variables as changed each time its driver started or its device reconnected, with nothing changed
- Fixed the Bluetooth Coordinator's Driver Status reading Running while the license had lapsed; it now reads No active license
- Fixed an ESPHome driver logging an error every 30 minutes while Automatic Updates was On, and waiting a little longer to reconnect a device each time
- Fixed button link LEDs staying lit when a light was bound to another device that reports off
- Fixed contact sensors and relays running their programming again when nothing had changed: after a driver start or a reconnect, or when a consumer is bound, only a real change now fires it. Bluetooth presence fires its Home and room Occupied events only for a real change too, and a device or room not heard again after a restart now reads away or empty once the Away Timeout passes
- Fixed passwords, keys and lock codes appearing in a driver's log at the higher log levels; they are now left out or masked at every level
- Fixed Reset Driver leaving part of a driver's setup behind: it now removes every connection, variable and event the driver made and puts its settings back to their defaults, keeping only what reaches the same device, its address and login. A device's encryption keys no longer go with it
- Fixed a Bluetooth Coordinator room deleting its Occupied and Empty events, and the programming on them, along with its connection and variables, when its last Bluetooth proxy was removed or moved to another room; the room now keeps them
- Fixed the Bluetooth Coordinator starting on a controller below its minimum OS; it now shows that it is disabled, as the other drivers do
- Fixed locks adding a history line and running their programming for a status that had not changed: at each status check, after a driver start, a reconnect or a bind, and when a lock or unlock was dropped for waiting too long. A fan no longer tells Control4 it turned on or off after a driver start or a reconnect when it had not
- Fixed a lock in Persistent mode going back to its old status for a while after a lock or unlock, with a history line for it, when its periodic status check came while the bolt was still moving
- Fixed a cover without position control on ESPHome 2025.8 or later reading unknown, with its Open and Closed contacts both closed, and ignoring Open, Close and Stop; its state and commands now use the position ESPHome reports and takes
- Fixed an ESPHome switch, climate device or water heater that had not reported a state yet reading as off, or with its default settings, on ESPHome 2026.9 or later; it now waits for the device's first state, as sensors do

Changed

- Changed how long the ESPHome driver waits without an active license before it drops its devices, from one minute to five, so a DriverCentral Cloud driver update or a slow controller start no longer drops a licensed system. A license reconnects the devices within 10 seconds
- Changed how long a color set on a light with Dim-To-Warm lasts: one set while the light is on now stays when the light is dimmed or brightened, until the light turns off, instead of until the next brightness command. The next time the light turns on, Dim-To-Warm applies again
- Changed Automatic Updates to one setting for the ESPHome driver and its sub-drivers: changing it on any of them now changes it on all of them, so the drivers update together
- Changed ESPHome Lock and ESPHome Yale to Composer's Locks category, where Control4's own lock drivers are listed
- Changed where each driver's own settings sit in Composer: Support Code and Cloud Monitoring now close Driver Settings, after the settings for the device

v20261027 - 2026-10-27

Added

- Added Button LEDs to the ESPHome driver: a button bound to a load lights one of the device's lights in the load's button colors, the way a Control4 keypad lights its buttons
- Added the ESPHome Light driver's button LED colors to Composer, for keypad buttons linked to the light

Fixed

- Fixed a keypad button linked to an ESPHome light showing the wrong LED state until the light next changed
- Fixed a Yale lock or unlock, or a SwitchBot press, running minutes after it was tapped on a slow connection; a command still waiting after 60 seconds is now dropped and reported as failed

v20261024 - 2026-10-24

Added

- Added the ESPHome SereneScent sub-driver for Homedics SereneScent BLE diffusers, with power, mist intensity and light color control, keypad button links and a power relay
- Added automatic offline key updates to the Yale driver: after Verify and Fetch Keys it checks the Yale cloud for a new key daily and whenever the lock rejects the key, so a changed key no longer means running the actions again

Fixed

- Fixed a Yale lock bound to the Bluetooth Coordinator failing every connection: the lock dropped the link at the handshake because its notifications were never turned on; devices that answer by notification now work through the coordinator as they do bound to a proxy
- Fixed a device connected through the Bluetooth Coordinator not noticing when its connection closed: a Yale lock that turned its key down waited out a 10-second timeout every time instead of handling it as a closed connection
- Fixed the SwitchBot and Yale drivers waiting indefinitely when the Bluetooth proxy or coordinator never finished a connection request, as when it restarted mid-connect: a SwitchBot stayed Busy with the command queued, and a Yale lock in Poll mode stopped polling; the proxy now fails a connection after 30 seconds and the coordinator tries the next proxy, and the driver still gives up after a little over three minutes if nothing answers
- Fixed a Yale lock in Persistent mode counting a failed or timed-out handshake as two failures, using up its reconnect attempts twice as fast and retrying a lock or unlock that hit it as a status read instead
- Fixed the SwitchBot and Yale drivers keeping the proxy's connection slot when a connection came up without the device's Bluetooth characteristics, as can happen on a first connect: a Yale lock stayed on "Error: Missing characteristics" and stopped polling or reconnecting; the driver now disconnects, frees the slot and handles it as a failed connection
- Fixed a SwitchBot Relay Switch with no encryption key staying Busy and keeping the proxy's connection slot when a connection came up without the device's Bluetooth characteristics; it now disconnects and frees the slot

- Fixed a SwitchBot Bot holding one of the ESPHome device's Bluetooth connection slots after it was bound, or after the device reconnected, until its next press; it now disconnects a few seconds after connecting
- Fixed a Yale lock abandoning its connection, and a SwitchBot showing Connection Failed for a press that went through, when a connection failure was reported after the device had connected
- Fixed one failed Bluetooth connection using up two of a Yale lock's reconnect attempts in Persistent mode, and taking a SwitchBot from Connection Failed straight to Listening
- Fixed a Yale lock dropping a lock or unlock tapped while it waited for its first connection after a restart or a change of Connection Mode; it now carries it out once connected
- Fixed a Yale lock newly bound to the Bluetooth coordinator staying on "Waiting for data" until it was locked or unlocked; it now connects on the lock's next advertisement
- Fixed a Yale lock keeping the proxy's connection slot and stopping polling or reconnecting when a notification subscription failed or went unanswered; the driver now disconnects, frees the slot and handles it as a failed connection
- Fixed the Bluetooth coordinator trying a proxy with no free connection slot, and a slow connection taking up to 90 seconds on one proxy and four and a half minutes through the coordinator; the coordinator now moves on from a full proxy at once, and one proxy's attempt takes about a minute at most
- Fixed a Bluetooth connection that failed or gave up leaving the ESPHome proxy's connection slot in use, which could keep other SwitchBot and Yale devices on that proxy from connecting; the proxy now disconnects and frees the slot
- Fixed Advanced Lighting scenes on an ESPHome light: a flash scene with no delay or ramp time hung the driver, Stop Scene Ramp did not stop the light, and a scene kept moving a light that was sent another command, or showed it ramping after the scene was deactivated; a step now starts where the light is, fades over its own ramp time and takes Color On Mode and Dim-To-Warm
- Fixed overlapping ramps, fades, scene steps and stops starting from, jumping to or being shown at the wrong level or color, or not stopping the light; a stop now holds the light where it is and lets a fade it did not stop finish
- Fixed a ramped On that changes a light's color starting in the old color, which kept some lights dark for most of the ramp, or restarting a moment in, and a color sent right after it being dropped
- Fixed a held Top or Toggle button ramping toward the Default On level instead of full, and a button pressed during a ramp acting on the level it began from

- Fixed a light ignoring its Color On Mode, on level and default ramp rate until one was changed in Composer, and Color On Mode not applying a color preset or the right Previous color; a color sent while the light is off or fading off now waits for its next on instead of turning it on or being lost
- Fixed a cold and warm white light coming on dimmer and warmer than asked with Dim-To-Warm, a Dim-To-Warm light being shown in two colors at every level change, and a light's On or Dim-To-Warm color being undone by a scene's color or one set while it was off
- Fixed ESPHome lights with a white beside their color, such as RGBW and RGBCT lights, keeping the white lit or shown under a color and the color lit under a color temperature, and making whites out of red, green and blue; a color at or near a white, such as a new light's On Color, now lights the white, and a color temperature beyond the light's range is shown as set
- Fixed Max On Level and Min On Level having no effect; a light's 1 to 100 % now runs between them, as on a Control4 dimmer
- Fixed a light's level or color set from elsewhere, such as Home Assistant, being undone when a Control4 ramp ended, or not shown while it was off
- Fixed a light sent a level and then a color, such as from Programming, being shown for a moment in the color it had before
- Fixed a light turned on over a ramp time being shown as off until the ramp ended; Control4 now shows it on from the start
- Fixed a light asked for its level during a ramp being shown as stopped where the ramp began, or as off in a ramp up from off
- Fixed the keypad LEDs on a light's button links showing it off while it was turned on over a ramp time, and staying lit when the light went offline
- Fixed a light turned on over a ramp time telling Control4 its Color On Mode's color twice, the first time without its color preset
- Fixed a light stopped or turned off during a ramp up from off saving 1 % as its Previous On level instead of the one it had, or the level it had reached
- Fixed a light sent a level and then another soon after, such as a stop just after a ramp up from off began, being shown for a moment at the first level, which the stop then saved as its Previous On level
- Fixed a light being shown as off, and running its Light Off Programming and then its Light On Programming, when Log Level, Log Mode or Device Log Forwarding was changed while the device's logs were forwarded, as when Log Mode turns itself off after three hours
- Fixed a light asked for its level before its ESPHome device had reported being shown as off

- Fixed a light running its Brightness Changed, Level Change or Color Changed Programming after a controller restart, and a light in a full color running its Color Changed Programming when its driver was updated, its ESPHome Light connection was bound or its device reported again, as Control4 was told its level and color again; it is now told only a level, color or capability that differs from the one it shows. After a controller restart Control4 still reports the color variables of a light with color or color temperature as changed, as it learns the light's capabilities again
- Fixed an on/off light being shown as off, and running its Light Off Programming and then its Light On Programming, when its driver was updated while it was on
- Fixed a light sent a command while its ESPHome device was offline, or its ESPHome Light connection was removed, being shown on, off or at the level sent, and running its Light On or Light Off Programming; it now ignores the command and shows only what the device reports
- Fixed a thermostat being shown in an HVAC mode, vane position or water heater mode sent while its ESPHome device was offline, or its ESPHome Climate connection was removed, which the device never took
- Fixed a light, fan or lock still being shown online after its ESPHome Light, ESPHome Fan or ESPHome Lock connection was removed
- Fixed a switch, number, select, text, date, time or datetime variable keeping a value programming set while the ESPHome device was offline, or a date or time in another format, which the device never took; it now goes back to the device's value
- Fixed the ESPHome driver's Driver Status reading Refresh failed for a few seconds, in place of why it was not connected, when a connection was made or a sub-driver loaded while the device was offline
- Fixed a device selected only under Select Presence Devices on the Bluetooth Coordinator never being placed in a room: it was only ever Home or Away, its room events and room occupancy never changed and its Present binding never closed; every presence device's room now comes from the signal each proxy reports for it, smoothed by the RSSI Smoothing Factor
- Fixed a presence device on the Bluetooth Coordinator staying in a room whose proxy no longer saw it, until it went Away, whenever the proxy that did see it was not stronger than its last signal there by the Room Change Hysteresis
- Fixed a Bluetooth Coordinator presence device's RSSI variable showing a long fraction of a dBm or, for a device selected only for presence, the raw reading from whichever proxy heard it last; it now shows the signal Minimum Room RSSI is checked against, in whole dBm
- Fixed a presence device on the Bluetooth Coordinator moving to another room on one weak signal from that room's proxy when the proxy had last heard it close by, more than 30 seconds before; that proxy's signal now starts again from what it hears

- Fixed a Bluetooth Coordinator presence device's Present binding staying open while the device was home but in no room, as below Minimum Room RSSI; it now closes when the device comes home
- Fixed the Bluetooth Coordinator's Last Presence variables still describing the previous event when Left Room and Any Device Left Room fired for a device whose signal dropped below Minimum Room RSSI
- Fixed a presence device on the Bluetooth Coordinator staying in a room after that room's proxy was disconnected from the coordinator, so the room showed empty with the device in it once the proxy was connected again; the device now leaves the room, stays Home and is placed again by its next signal
- Fixed a room on the Bluetooth Coordinator staying Occupied, with the device still counted, after a presence device in it was removed from Select Presence Devices, so later occupants never fired its Occupied or Empty events; the room now empties when its last device is removed
- Fixed a light being shown as off, and running its Light Off Programming, when the DriverCentral license lapsed; it now shows offline at the level it had, and a command sent meanwhile does not change it, until the license is active again and the light reports its level, while a light whose ESPHome device drops off the network is still shown as off
- Fixed the Bluetooth Coordinator marking its presence devices Away, and running their Away, Left Room and Room Empty Programming, when the DriverCentral license lapsed
- Fixed the ESPHome driver disconnecting its device at a controller start, before the DriverCentral Cloud driver had answered, which showed a light that was on as off and ran its Light Off and then its Light On Programming; it now stops only once a minute passes without an active license, and connects again as soon as one is active
- Fixed Yale locks failing almost every connection since v20260802, so Poll mode never read the lock and Persistent mode gave up after its retries
- Fixed a Yale lock showing an unknown state for a moment after every lock and unlock; a lock or unlock that jams now shows a fault
- Fixed a Yale lock showing as unlocked for a moment just after it locked, and as locked just after it unlocked
- Fixed Verify and Fetch Keys taking another lock's key in a Yale account with several locks when this lock's MAC address was not known yet
- Fixed Yale cloud errors showing in Yale Cloud Status as "Error: table" instead of the error
- Fixed the Yale driver writing the lock's offline key to the log at Debug level when fetching keys

- Fixed a SwitchBot Relay Switch ignoring the device's reply to every command, so a command it refused, as with an incorrect encryption key, went unreported
- Fixed the Programming events of an ESPHome event entity, a BTHome button or dimmer and a Govee probe or sensor error leaving the device's name out of their script title in Composer Pro, as in "Button pressed once"; the title now names the device
- Fixed a Yale lock or unlock sent while the driver was still connecting being lost, with "Connection failed: Superseded by a newer request" in the log, and a lock or unlock retried after a failed connection undoing one sent since

Changed

- Changed how long a color set on a light with Dim-To-Warm lasts: it now lasts until the next brightness command, even one for the level the light is already at, instead of until the light is turned off

v20260926 - 2026-09-26

Fixed

- Fixed event, BTHome and SwitchBot button links not activating Advanced Lighting scenes; loads bound to them already worked
- Fixed entities with no name of their own, such as a relay configured with `name: None`, being left out or given a blank name; they now take their device's name
- Fixed entities that share an ESPHome key, such as a sensor and a text sensor with the same name, being set up as one entity; each now gets its own variables and connections
- Fixed commands to entities on an ESPHome sub-device being ignored on ESPHome 2025.8 and newer
- Fixed an event entity's Programming events disappearing after a controller reboot, or keeping their old name after a rename

v20260922 - 2026-09-22

Added

- Added presets to the climate driver, each storing a setpoint, HVAC mode, fan mode and vane position, and applicable from Navigator or from Programming
- Added preset scheduling by weekday and time; the schedule is kept by Control4 and the driver applies each preset as it falls due

- Added holds, so changing the thermostat or choosing a preset by hand holds that setting until the next scheduled event releases it; clearing a held preset sooner returns to the preset the schedule has in force, and a Permanent hold, offered with or without a schedule and released by no scheduled event, stays until cleared by hand or from Programming
- Added vane control in the Extras tab for climate devices that report swing modes
- Added a button connection for each event type an ESPHome event entity declares, so a touch button or gesture on the device can drive a light, scene or any other load without Programming
- Added a Bluetooth Proxy Status warning when the BLE scanner has stopped reporting devices and the driver has run out of ways to restart it, so a proxy that needs a power cycle no longer reads as healthy

Fixed

- Fixed lights, thermostats, water heaters, fans and locks still showing as connected when they were not: after the driver's IP address, port or credentials were changed or cleared, and, for thermostats and water heaters, after the ESPHome device itself went offline; they now go offline with the device until it reconnects
- Fixed thermostats and water heaters always showing Fahrenheit; they now follow the project's temperature scale and pick up a change made in Navigator straight away, with the Celsius/Fahrenheit setting in Composer available to override it for an individual thermostat
- Fixed a thermostat staying on its last mode after the unit was turned off outside Control4, for example with its own remote; it now shows Off, a unit that reports whether it is heating or cooling shows when that stops, and switching the fan to On is now shown too
- Fixed a thermostat never showing whether it is heating, cooling, idle, drying or running the fan only
- Fixed Heat never engaging on a water heater that had not yet stored an operating mode
- Fixed a temperature of exactly 0°C, or a humidity of 0%, not being shown
- Fixed a temperature, setpoint or humidity the device has not measured yet being shown as a number near 5.1e38, which a setpoint nudge then clamped to the maximum; such a reading is now left blank until a real one arrives, and sensor values are handled the same way
- Fixed a newly installed climate driver forwarding a bound temperature sensor's readings to the device before the thermostat had enabled the remote sensor

- Fixed the climate driver's temperature and humidity outputs appearing as audio connections in Composer instead of control connections; the humidity output moves to a different connection as a result, but nothing needs reconnecting because the audio one could not be connected to anything
- Fixed an "Error setting default color rate from driver" message in Composer when opening the properties of an ESPHome light
- Fixed Composer's test panel graying out the dimming controls for ESPHome lights that do support brightness; dimming from Navigator was never affected
- Fixed a BTHome or SwitchBot button doing nothing to a dimmer it was linked to, where sending the press, click and release together meant the release stopped the dim the click had just started; a button now reports a press followed by a click, the same as a Control4 keypad, so linked dimmers, switches and other loads all respond
- Fixed a SwitchBot Bot running its action twice for a single press of a linked keypad button, which turned a toggle back to where it started
- Fixed Bluetooth proxy recovery looking for a button with "restart" in its name to clear a stalled scanner, so a proxy that names its button differently or exposes none at all had no recovery, and where a button was found the whole device was rebooted, dropping every Bluetooth connection it held; recovery now restarts the scanner itself on any proxy that reports its scanner state, leaving those connections up

Changed

- Changed the setpoint display for climate devices that report a single target: they now show one setpoint instead of a heat and cool pair, since most heat pumps and mini splits hold one target and decide internally whether to heat or cool toward it, so the pair could never be honored; Auto is unaffected
- Changed the documentation to cover the `Connected` variable that every driver publishes, so it can be used in Programming to show whether a device is online

v20260816 - 2026-08-16

Added

- Added the konnected Smart Garage Door Opener to the verified devices, and generalized the ratgdo setup into a shared Garage Door Configuration guide that covers both

Fixed

- Fixed Bluetooth proxies logging a spurious "Property 'Select Bluetooth Devices' not registered" warning at every driver startup
- Fixed the Bluetooth device selection being permanently erased when a proxy was bound to a Bluetooth Coordinator; it is now kept and restored if the proxy is later unbound
- Fixed Bluetooth proxies in Coordinator Mode reporting their connection slots as "(Oversubscribed)"

v20260802 - 2026-08-02

Added

- Added a `Connected` (BOOL) variable to the main driver and every sub-driver with a Driver Status so Programming can react to connect/disconnect
- Added an "Open Latch" action to the ESPHome Lock driver for locks that support the open command (`supports_open`)

Fixed

- Fixed a slow TCP connect interfering with the connection attempt that replaced it, which could run the handshake on the wrong socket and tear down a healthy connection when the stale socket closed
- Fixed the HVAC state freezing on the previous value during a heat pump's defrost cycle; defrosting now reports as Heating, and unmapped climate modes/actions log a warning instead of being silently dropped
- Fixed latch-style ESPHome locks (`supports_open`) reading as "unknown" in Control4 while reporting the open or opening state; both now map to unlocked
- Fixed overlapping status refreshes silently racing each other for the same response callbacks; refreshes are now serialized
- Fixed cover contacts and Yale DoorSense sending nothing to a newly bound consumer (or after a DoorSense drop and re-detect) until the state changed; consumers are now seeded with the last known state on bind
- Fixed the Reset Driver action leaving bound consumers stale until the next state change (sensor values, cover contacts, BTHome bindings, Yale DoorSense)
- Fixed pending requests (refresh, Bluetooth GATT operations) hanging forever when the connection dropped mid-request; they now fail immediately with a "Disconnected" error

- Fixed SwitchBot channel relays and contact sensors (motion, contact, leak, tamper) leaving bound consumers stale after a driver restart until the next state change
- Fixed a pending request hanging forever when a second request for the same response type started before it finished; the older request now fails with a "Superseded by a newer request" error instead of stalling, and the newer request is no longer left without its callbacks (which could make entity discovery or Bluetooth GATT service discovery return an empty result)
- Fixed a superseded status refresh being reported as "Refresh failed" and dropping the device connection out from under the refresh that replaced it

v20260728 - 2026-07-28

Added

- Added `TEMPERATURE_VALUE` and `HUMIDITY_VALUE` connections for sensors with a `temperature` or `humidity` device class, so they can be bound to thermostats and other value consumers; the reported scale honors the sensor's declared unit of measurement

Fixed

- Fixed cover contacts, Yale DoorSense, BTHome bindings, and the SwitchBot Bot relay leaving bound consumers stale after a driver restart or update until the next state change
- Fixed all entity states appearing frozen in Control4 (covers stuck on "Unknown", sensor variables never updating) on ESPHome 2026.7+ firmware, which no longer sends the deprecated `object_id` field; the driver no longer reads `object_id`, and log messages now identify entities as `type 'Name' (key=N)`
- Fixed devices logging `using outdated API 1.0, update to 1.14+` on every connection; the ESPHome native API version advertised in `HelloRequest` is now 1.14 rather than 1.0

v20260711 - 2026-07-11

Fixed

- Fixed Bluetooth Coordinator connections to ESPHome proxies (and other dynamically created connections) disappearing after a controller reboot or Director restart; dynamic bindings are now restored early enough in driver startup for Director to reconnect them

- Fixed an on/off-only ESPHome light advertising brightness and color to capability consumers that read the static declaration instead of the runtime-narrowed set; hardware capabilities (dimming, color, color temperature) are now declared conservatively in the static baseline and enabled at runtime from the entity's discovered color modes
- Fixed BOOL variables (`<Entity> State` for binary_sensor and switch, plus all BTHome boolean sensors) staying as `False` in the Variables Agent even when the underlying state was changing; variables now serialize as `"0"` / `"1"` matching what Control4 expects
- Fixed ESPHome fan `Designate Preset` command: the handler now reads the correct `PRESET` param (was `SPEED`), clamps to the driver's speed count, persists the value across driver restarts, notifies the proxy so Composer and Navigator reflect the designated preset, and applies the preset when the fan is turned on so `Turn On Fan` runs at the designated speed

v20260512 - 2026-05-12

Added

- Added brightness and dimming support to ESPHome lights with smooth ramping, preset management, and hold-to-dim button control
- Added color and color-temperature support to ESPHome lights for every ESPHome color mode (white, color-temperature, cold/warm white, RGB, RGBW, and combined RGB + white modes)
- Added Advanced Lighting Scenes support to ESPHome lights so they can participate in lighting scenes alongside other Control4 dimmers

v20260418 - 2026-04-18

Added

- Added Event entity support: stateless triggers (button presses, gestures, doorbell rings) now create Control4 events for programming and track the last event type in a variable
- Added Date, Time, and Datetime entity support: configurable date/time values on the device are exposed as writable string variables (YYYY-MM-DD, HH:MM:SS, YYYY-MM-DD HH:MM:SS)
- Added Climate entity support: ESPHome climate devices are exposed as thermostatV2 sub-drivers with HVAC mode, setpoints, fan mode, presets, and humidity control

- Added Select entity support: STRING variable with the current option, writable via programming or variable writes
- Added "Set Select" programming command with dynamic Select and Option dropdowns

v20260325 - 2026-03-25

Fixed

- Fixed cover contact sensors sending duplicate notifications during open/close operations
- Fixed Yale DoorSense contact sensor sending duplicate "Closed" notifications on every poll cycle by tracking the last known door status and only reporting on actual state changes

v20260319 - 2026-03-19

Fixed

- Fixed an issue where entities were no longer being detected reliably on connection

v20260318 - 2026-03-18

Fixed

- Fixed Bluetooth Coordinator failing to connect to active BLE devices (SwitchBot, Yale locks) through proxies

v20260314 - 2026-03-14

Added

- Added fan support with on/off, speed control (1-6 speed variants), direction, and oscillation
- Added ESPHome Yale sub-driver for Yale/August BLE smart locks with lock/unlock control, door sense, and battery monitoring

v20260217 - 2026-02-17

Added

- Added Bluetooth proxy support with scanner infrastructure, advertisement parsing, and GATT connection management
- Added ESPHome Bluetooth Coordinator driver for multi-proxy aggregation with RSSI-based routing and connection failover
- Added room presence tracking with RSSI-based detection, anti-flapping, and contact sensor bindings
- Added ESPHome BTHome sub-driver for Shelly BLU and BTHome v1/v2 sensors
- Added ESPHome Govee sub-driver for temperature, humidity, and meat thermometer sensors
- Added ESPHome SwitchBot sub-driver for Bot, Plug Mini, Meter, Motion, and Contact devices
- Added device log forwarding to the ESPHome driver

v20251031 - 2025-10-31

Fixed

- Fixed compatibility with ESPHome 2025.10.0 for devices configured without passwords
- Fixed password authentication failures being poorly detected and reported

v20251022 - 2025-10-22

Fixed

- Fixed an issue with parsing unknown fields in protobuf messages

v20251019 - 2025-10-19

Added

- Added support for OpenSSL with "Encryption Key" authentication mode across all applicable algorithms

Fixed

- Fixed a bug with the authentication flow in the latest 2025.10.0 firmware

v20250811 - 2025-08-11

Fixed

- Fixed switch entities not responding to bound relay proxies

v20250715 - 2025-07-14

Fixed

- Fixed bug causing entities to not be discovered on connect

v20250714 - 2025-07-14

Added

- Added support for encrypted connections using the device encryption key

v20250619 - 2025-06-19

Added

- Added ratgdo specific documentation

v20250606 - 2025-06-06

Added

- Initial Release