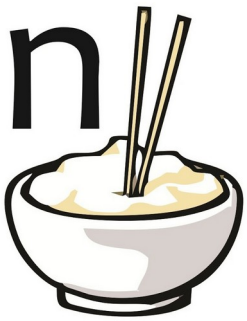
chowmain
software & apps



Bond Bridge RTI Driver

bond Bridge

Revision: 20200110
Date: 2020/01/10
Author(s): Richard Woodburn

Contents

Bond Home

This is a mobile application by Olibra LLC that can be downloaded from the mobile device application store.

Important: All supported actions need to be first configured in the mobile app before it can be interpreted from the Bond Driver.

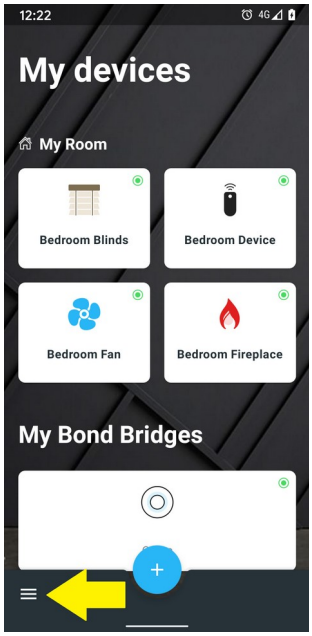
Settings for Bond Driver

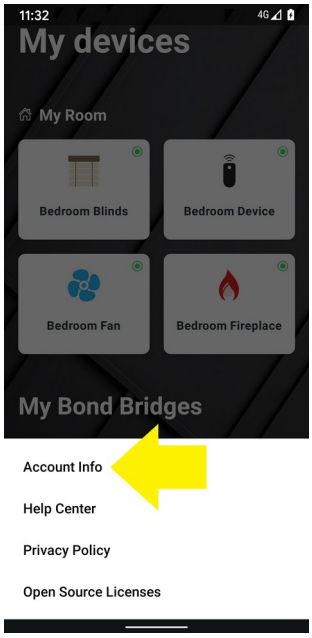
Once the mobile app has been set up with all the required devices and commands, information for the setup of the bond driver can be found here.

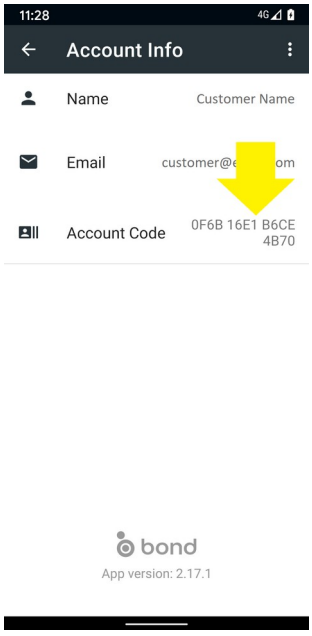
Account Code

Steps to find the account code used in the bond device.

1. Click the menu icon at the bottom left of your screen.
2. Click on the Account Info menu item
3. Make note of the 16 digit code on the Account info page.



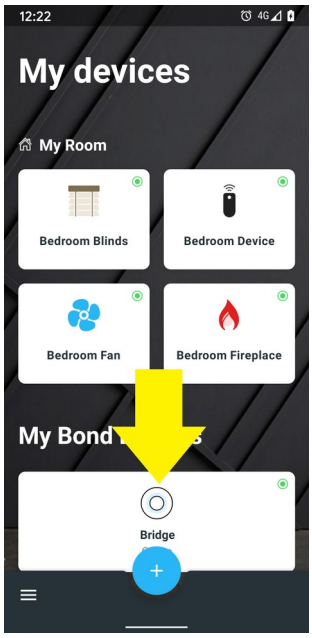


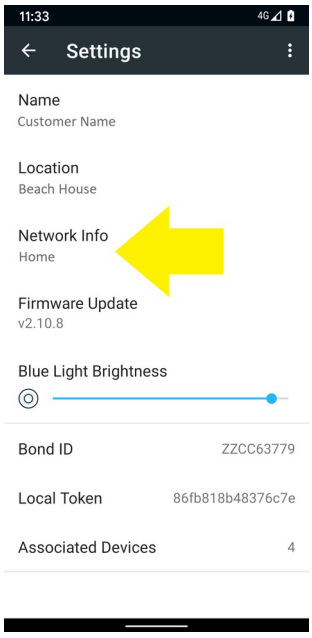


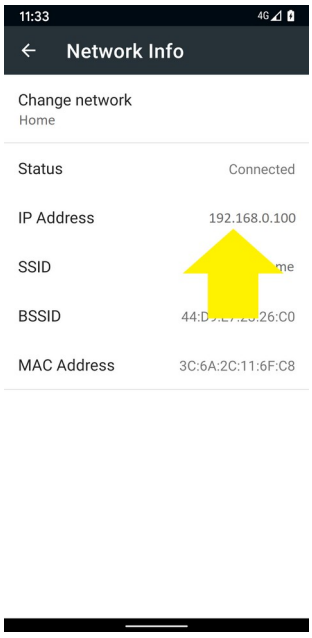
IP Address

Steps to find the account code of the bond device.

1. Click the Bond Bridge device
2. Click Network Info
3. Write down the Ip Address of the bond device.





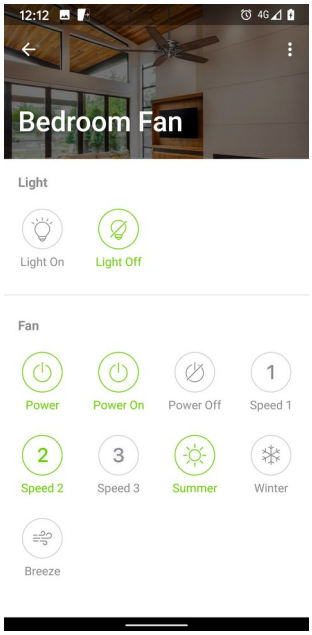


Actions for Bond Driver

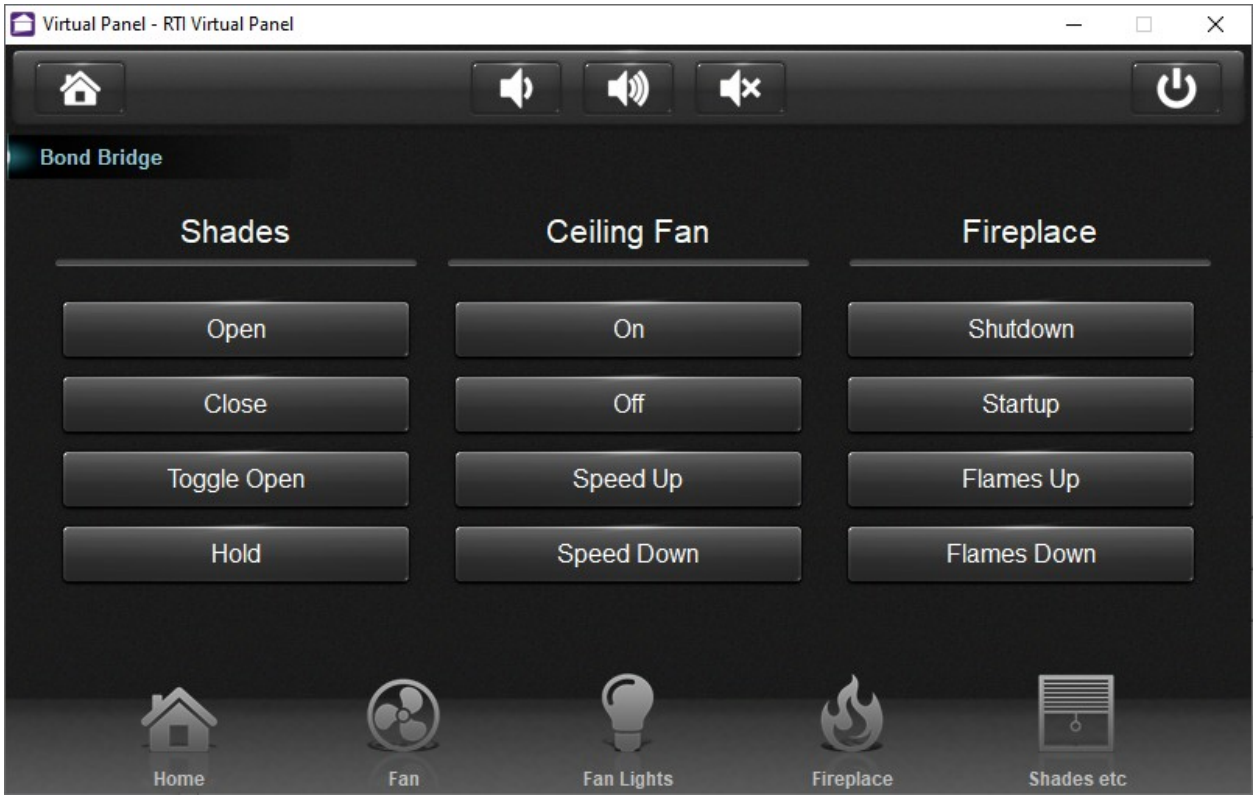
When setting up macros in RTI many actions will be available to select. It can be helpful to note the available actions so as to not select an action that bond will ignore.

Test and confirm that the action works in the bond app before selecting from the bond driver.

The example project that comes with this driver shows many examples of how to call and pass parameters to actions.



Driver Overview



The Bond Bridge RTI driver allows for control of the Bond Bridge device and stand alone Bond Smart devices from the RTI system.

The driver allows for many actions to be performed on Ceiling Fans, Motorized Shades, Electric fireplaces and Other Generic Devices like Garage Doors.

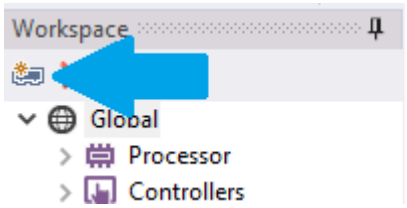
Driver Installation

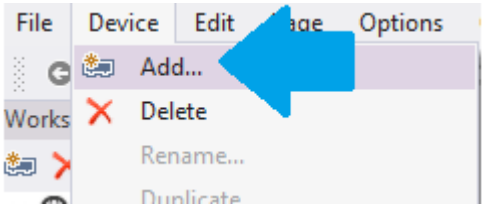
The zip file that included this documentation has the rtidriver file you will need to add. The first step is to download and extract the driver from the zip file. It doesn't matter where you store the file but we advise keeping them together.

4. The default location is Documents\Integration Designer\Control Drivers

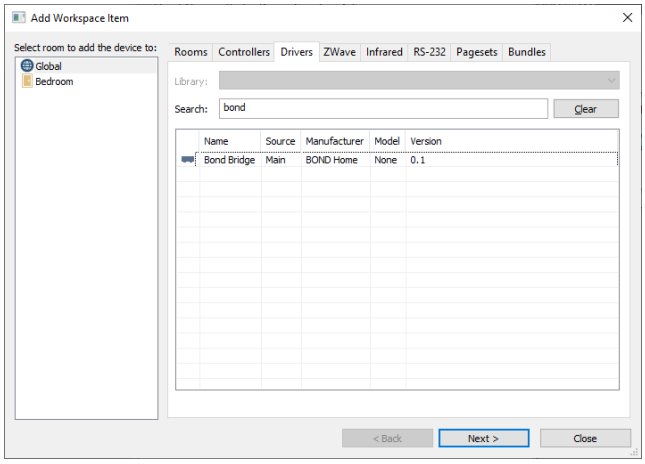
Add the driver

Select Add Workspace Item from the top of the Workspace panel or Add.. from the Device menu



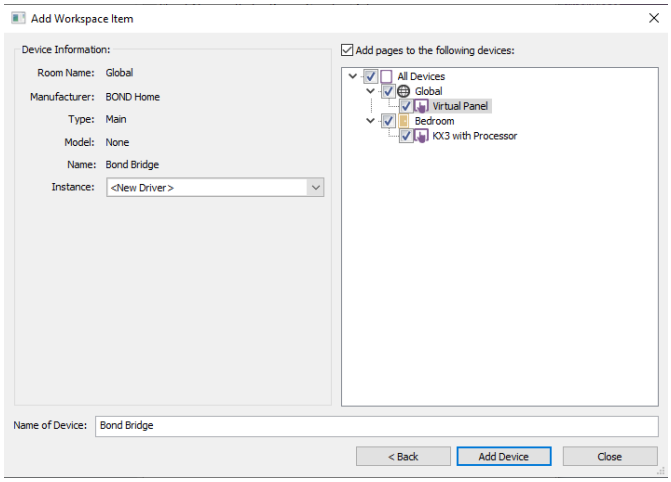


Click on the Drivers Tab and select Bond Bridge by Bond Home from the list of Drivers. You can also enter bond in the search bar to help with selection.



Select the location of the driver, then click Next > at the bottom of the dialogue.

Review Device information, Add pages to devices as required and Name the device then click Add Device.



The driver is now ready to configure.

Update an existing driver

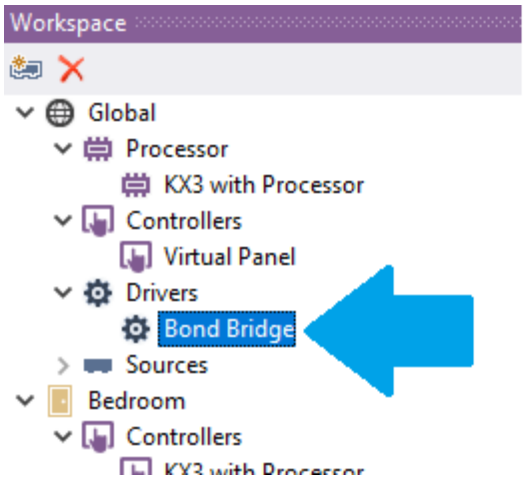
It is recommended that you copy the updated driver to your default driver folder.

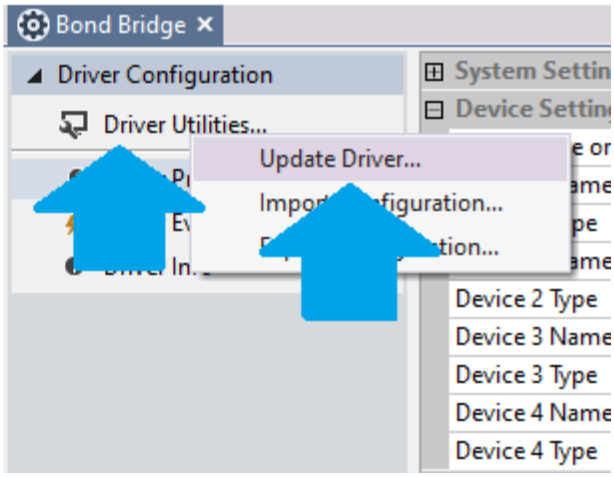
This will prevent accidentally selecting an old driver when starting a new project.

5. The default location is Documents\Integration Designer\Control Drivers

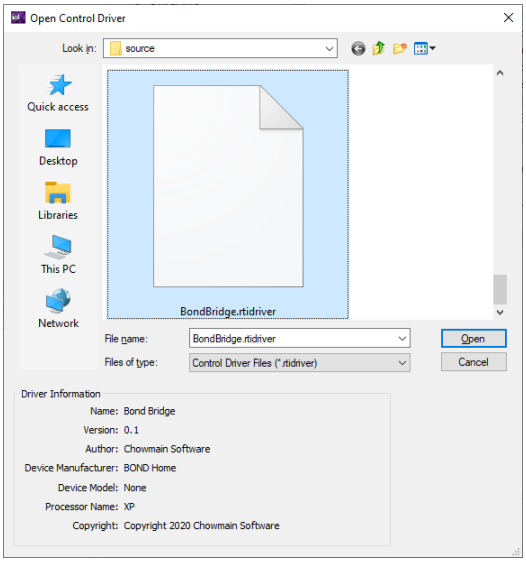
Once the file has been replaced, it will need to be updated in the project.

Select the Bond Bridge driver in your workspace then Click on Driver Utilities… and Update Drivers…





Navigate to the driver location select and click open.



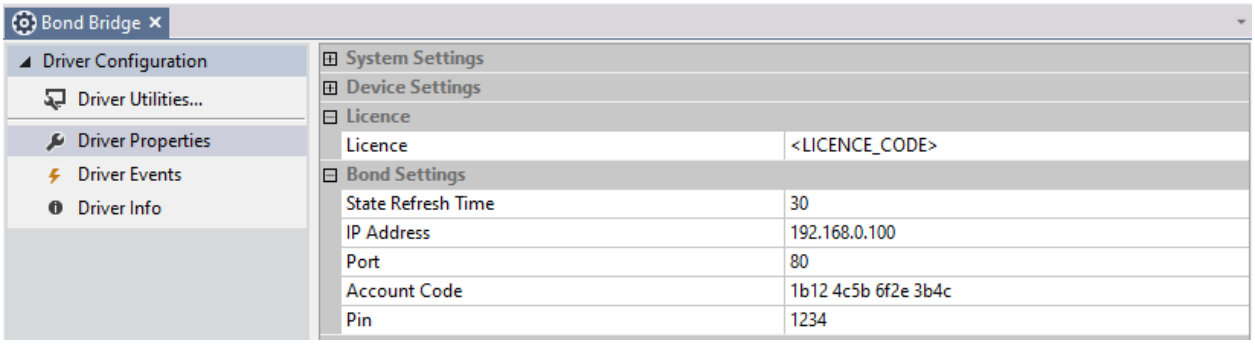
Note: You can review driver information in the open control driver dialogue.

Driver Configuration

From the Workspace expand and select Drivers > BondBridge

Next click on Device Properties

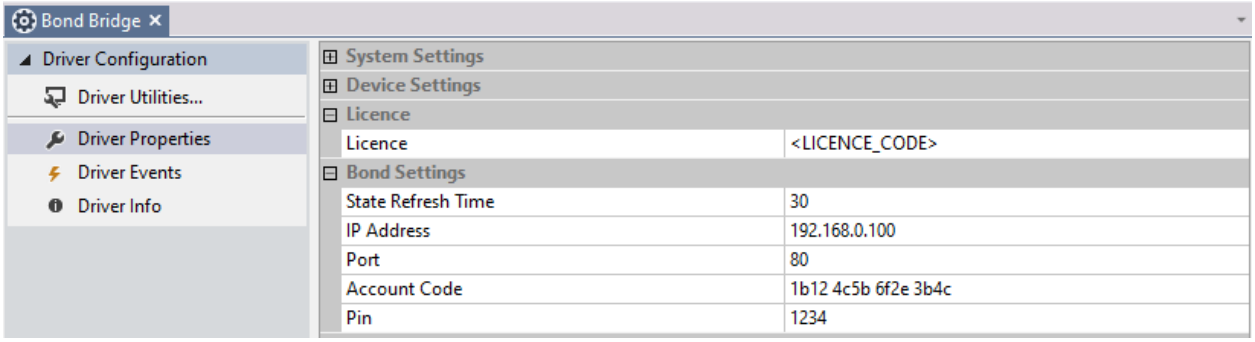
Licence



The driver will work without a licence for 7 days, automatically entering the trial phase if you don't enter a licence key. To keep using the driver after the trial has expired you will need to purchase a licence key.

Once you have your key it should be entered into the Licence field.

Bond Settings



State Refresh Time

This is a timer value where the driver will ask bond for the current state of each device mapped into RTI.

A check on the bond token and device list is also made at this time, if either are removed through the device commands the driver will attempt to update during this time.

IP Address

The IP Address of the bond bridge or stand alone bond smart device can be found in the bond mobile app.

Port

The Port default is set to 80 and should not be altered unless advised.

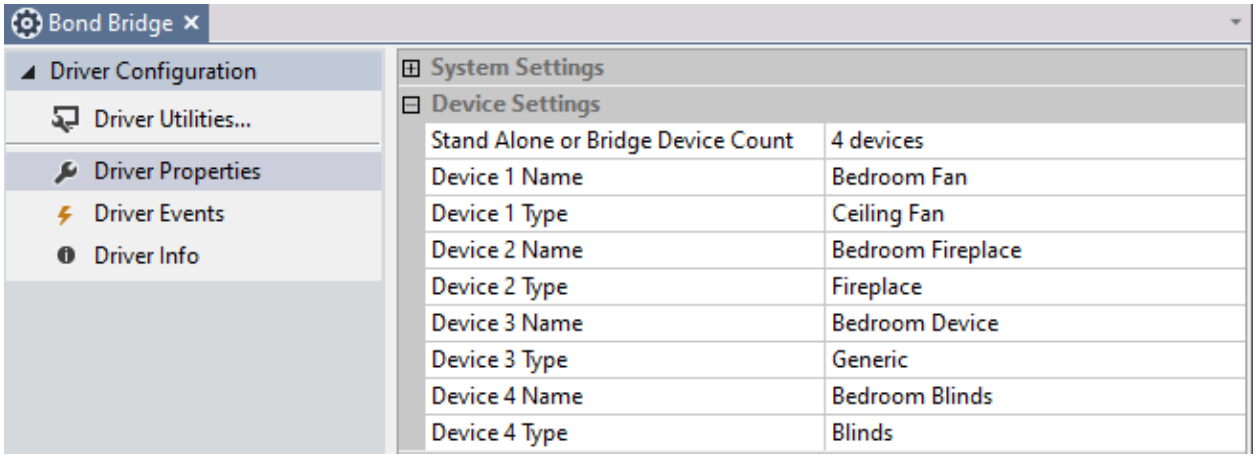
Account Code

The account code can be found in the bond mobile app by selecting the menu icon > account info. Enter the code with spaces included.

Pin

The Pin is found on the bottom bond bridge and is 4 digits in length.

Device Settings



Stand Alone or Bridge Device Count

The driver can connect to either a Bond Bridge or to a stand alone bond device, a separate driver instance is required per stand alone and bridge device.

If you select stand alone only the type of the individual device is required. Otherwise select the count of devices you wish to map from the bond device.

Note: Not all devices are required to be mapped and any found on the bridge not listed here will be ignored by the driver.

Device X Name

The name of the device as it appears in the bond mobile app. This name must be unique per device on the bond bridge and must match the name provided here.

This information is not required for stand alone bond smart devices.

Device X Type

The type of the device as it appears in the bond mobile app. This type is what determines which actions will be available in RTI to call.

Note: A device must support an action before it will be called from the driver, typically this will be the actions that are mapped in the bond mobile app.

The driver is now ready to use.

Driver Variables

Licence Valid

6. licenceValid: (boolean) true if the driver licence is valid, false if not valid

Licence Info

- licenceInfo: (string) description of the current licence state

Token Valid

- tokenValid: (boolean) true if a token was successfully obtained from the bond device.

Note: if this is still false after driver loading is complete please check the following.

- IP Address is correct
- Account Code and Pin is correct

  or

- Perform a proof of ownership on the bond device
  - If a bridge device this is done by unplugging and plugging the device from its power source
  - If a smart device like a ceiling fan this is done by holding the power on the remote for 5 seconds

Device Map Count

4. deviceCount: (integer) count of the devices found on the bond bridge that have been mapped to the devices named in the driver configuration.

Note: if this number is less than expected please check the driver settings for:

- The device count is correct
- Each of the device names are correct (check with the mobile app)
- There are no blank or empty devices.

Driver Loading

- driverInit: (boolean) this is set to true while the driver is initializing and set to false once completed, other actions to the bond driver should not be performed during this phase.

Driver Commands

Get State

- getDriverDetails: requests detailed system information to be sent to the log.

Remove Token

- removeToken: removes the currently held bond token from memory, this can be required if switching bridge devices without setting up a new driver.

Remove Devices

5. removeDevices: removes the currently loaded devices from memory, the device list is collected on every startup of the driver, this can be run if changes have been made on the mobile app.

Get Device List

7. getDeviceList: requests all mapped device information to be sent to the log.

Bond State and Actions

Power

The Power feature controls the basic on/off state of a device.

For Ceiling Fans, it refers to the state of the fan motor.

For Fireplaces, power refers to the state of the flame.

State Variables

- power: (integer) 1 = on, 0 = off

Actions

- TurnOn: Turn device power on.
- TurnOff: Turn device power off.
- TogglePower: Change device power from on to off, or off to on.

Notes

- Most ceiling fans have lights which are not governed by the Power feature.
- Many fireplaces have separate light or fan functions, which are not governed by the Power feature

Timer

The Timer feature allows turning off a device after a specified delay, similar to the dial timer interface on toaster ovens.

State Variables

- `timer`: (integer) seconds remaining on timer, or 0 meaning no timer running

Actions

- `SetTimer`: (integer) Start timer for x amount of seconds. If power if off, device is implicitly turned on. If argument is zero, the timer is canceled without turning off the device.

Notes

8. The `Timer` feature requires the `Power` feature.
9. The timer is canceled implicitly by any action on the `Power`, `Speed`, or `Breeze` features, other than `TurnOn`. For example, if a timer is running, and the user turns off the device and then turns it back on, the timer will be canceled and therefore the device will not turn off again unexpectedly.
10. The intention that a timer is designed to help reduce energy consumption, but should never surprise the user who forgot that they enabled the timer function earlier. When the timer reaches zero it runs `TurnOff`, so it will turn off the device whether it is set at a specific speed or it is set to breeze.

Speed

The Speed feature is used by multiple-speed Ceiling Fans to track the motor speed.

State Variables

- speed: (integer) value from 1 to max_speed. If power=0, speed represents the last speed setting and the speed to which the device resumes when turned on.

Actions

- SetSpeed: (integer) Set speed and turn on. If speed>max_speed, max_speed is assumed. If the fan is off, implicitly turn on the power. Setting speed to zero or a negative value is ignored.
- IncreaseSpeed: (integer) Increase speed of fan by specified number of speeds. If the fan is off, implicitly turn on the power.
- DecreaseSpeed: (integer) Decrease fan speed by specified number of speeds. If attempting to decrease fan speed below 1, the fan will remain at speed 1. That is, power will not be implicitly turned off. If the power is already off, DecreaseSpeed is ignored.

Notes

- The Speed feature requires the Power feature.
- While many Fireplaces have a built-in fan, they do not use the Speed feature. See FpFan feature.
- When the device is turned off, the previous speed is remembered. When the fan is then turned back on, it will resume at the previous speed.

Breeze

The Breeze feature of many multi-speed Ceiling Fans provides a randomized breeze.

Breeze works by pseudorandomly changing the power and speed of the fan over time to create a natural breeze effect. There are two parameters of the breeze which may be adjusted to provide the desired breeze effect.

State Variables

- breeze: (array) array of the form [<mode>, <mean>, <var>]:
    - mode: (integer) 0 = breeze mode disabled, 1 = breeze mode enabled
    - mean: (integer) sets the average speed. 0 = minimum average speed (calm), 100 = maximum average speed (storm)
    - var: (integer) sets the variability of the speed. 0 = minimum variation (steady), 100 = maximum variation (gusty)

Actions

- BreezeOn: Enable breeze with remembered parameters. Defaults to [50,50].
- BreezeOff: Stops the breeze. Fan remains on at its current speed.
- SetBreeze: (integer, integer) Enable breeze with specified parameters
    - First parameter: Is for mean/average speed 1-100.
    - Second parameter: Is for the variability of the speed 1-100.

Notes

- The Breeze feature requires the Speed feature.
- SetSpeed implicitly disables breeze mode.
- mode: is always set to 1 when SetBreeze is called

Direction

The Direction feature is used by reversible Ceiling Fans to track the direction of the fan motor.

State Variables

- direction: (integer) 1 = forward, -1 = reverse.

Actions

- SetDirection: (integer) Control forward and reverse.
- ToggleDirection: Reverse the direction of the fan.

Notes

6. The Direction feature requires the Power feature

Light

The Light feature governs the basic on/off status of a device's main light.

This is a very common feature of Ceiling Fans, and present on many Fireplaces.

State Variables

- light: (integer) 1 = light on, 0 = light off

Actions

- TurnLightOn: Turns the light on.
- TurnLightOff: Turns the light off.
- ToggleLight: Switch the light from on to off, or off to on.

Notes

- See the UpDownLight feature for the behavior of devices with dual lights.

UpDownLight

The UpDownLight feature governs the on/off status of a device's upwards- and downwards-facing lights, such as the ceiling-wash "up light" and direct "down light" found on some high-end ceiling fans.

The corresponding physical remote often has separate buttons for the UpLight and DownLight, but no button for just "Light". However, Bond always makes the Light feature available along with UpDownLight to make these devices easy to integrate. For example, saying "Alexa, Turn on the Light" corresponds to the TurnLightOn action, which will have a reasonable result for devices with UpDownLight.

State Variables

- up_light: (integer) 1 = up light enabled, 0 = up light disabled
- down_light: (integer) 1 = down light enabled, 0 = down light disabled

Actions

- TurnUpLightOn: Turn up light on.
- TurnDownLightOn: Turn down light on.
- TurnUpLightOff: Turn off up light.
- TurnDownLightOff: Turn off down light.
- ToggleUpLight: Change up light from on to off, or off to on.
- ToggleDownLight: Change down light from on to off, or off to on.

Notes

- If both up_light and light are 1, then the up light will be on, and similar for down light.
- Both up_light and down_light may not be simultaneously zero, so that the device is always ready to respond to a TurnLightOn request.
- TurnLightOff/TurnLightOn honor the up_light and down_light enable variables.
- That is, the user is able to use the factory remote to select a prefered combination of up and down light, and that combination is restored when TurnLightOn is called, perhaps through a voice integration.

Brightness

The Brightness feature governs lights which can be dimmed to specified brightness level.

This feature is common on classic Ceiling Fans whose remotes have displays. Note, however, that classic Ceiling Fans whose remotes do not have displays typically only support HoldToDim or HoldToDimUpDown feature.

State Variables

- `brightness`: (integer) percentage value of brightness, 1-100. If light=0, brightness represents the last brightness setting and the brightness to resume when the user turns on light. If a fan has no dimmer or a non-stateful dimmer, brightness is always 100.

Actions

- `SetBrightness`: (integer) Set the brightness of the light to specified percentage. Value of 0 is ignored, use TurnLightOff instead.
- `IncreaseBrightness`: (integer) Increase brightness of light by specified percentage. If the light is off, it will be turned on at (0 + amount).
- `DecreaseBrightness`: (integer) Decrease light brightness by specified percentage. If attempting to decrease brightness below 1%, light will remain at 1%. Use TurnLightOff to turn off the light. If the light is off, the light will remain off but the remembered brightness will be decreased.

Notes

- The brightness level is remembered on `TurnLightOff` and restored on `TurnLightOn`.

UpDownBrightness

The UpDownBrightness feature extends the Brightness feature to cover the ability of ceiling fans with separately dimmable up and down lights.

This feature is almost only found on Smart by Bond Ceiling Fans.

State Variables

- up_light_brightness: (integer) percentage value of up light brightness, 1-100.
- down_light_brightness: (integer) percentage value of down light brightness, 1-100.

Actions

- SetUpLightBrightness: (integer) Similar to SetBrightness but only for the up light.
- SetDownLightBrightness: (integer) Similar to SetBrightness but only for the down light.
- IncreaseUpLightBrightness: (integer) Similar to IncreaseBrightness but only for the up light.
- IncreaseDownLightBrightness: (integer) Similar to IncreaseBrightness but only for the down light.
- DecreaseUpLightBrightness: (integer) Similar to DecreaseBrightness but only for the up light.
- DecreaseDownLightBrightness: (integer) Similar to DecreaseBrightness but only for the down light.

Notes

- The brightness level of each light is remembered on TurnLightOff, TurnUpLightOff, TurnDownLightOff and restored on TurnLightOn, etc.
- IncreaseBrightness and DecreaseBrightness operate on whichever of the up and down lights are enabled, but will never enable or disable one or the other light.

Flame

The Flame feature is used by fireplaces to indicate flame level.

State Variables

- flame: (integer) value from 1 to 100. If power=0, flame represents the last flame setting and the flame to which the device resumes when the user asks to turn on.

Actions

- SetFlame: (integer) Set flame and turn on. If flame>100, 100 is assumed. If the fireplace is off, implicitly turn on the power. Setting flame to zero or a negative value is ignored.
- IncreaseFlame: (integer) Increase the flame level of the fireplace by a specified number of flames. If the fireplace is off, implicitly turn on the power.
- DecreaseFlame: (integer) Decrease flame level by specified number of flames. If attempting to decrease the fireplace below flame level 1, the fireplace will remain at flame level 1. That is, power will not be implicitly turned off. If the power is already off, DecreaseFlame is ignored.

Notes

- The Flame feature requires the Power feature.

Open

The open feature is used to describe a device that can be opened and closed. Common use cases are motorized shades and garage doors.

State Variables

- open: (integer) 1 = open, 0 = closed

Actions

- Open: Open the device.
- Close: Close the device.
- ToggleOpen: Close the device if it's open, open it if it's closed

Notes

- If your remote has a discrete stopping command, consider using the Hold action to stop the motion of the device.

FpFan

The FpFan feature controls a fireplace fan. The FpFan feature is independent of the power feature, which for fireplaces indicates whether the flame is on or off.

State Variables

- fpfan_power: (integer) 1 = on, 0 = off
- fpfan_speed: (integer) from 1-100

Actions

- TurnFpFanOff: Turn the fireplace fan off
- TurnFpFanOn: Turn the fireplace fan on, restoring the previous speed
- SetFpFan: (integer) Sets the speed of the fireplace fan

Misc, including dimmers

Collected here are some actions that may be used with other features, but have no state-change behavior on the Bond.

Actions

- `Stop`: This action tells the Bond to stop any in-progress transmission and empty its transmission queue.
- `Hold`: Can be used when a signal is required to tell a device to stop moving or the like, since Stop is a special "stop transmitting" action
- `StartDimmer`: Start dimming. The Bond should time out its transmission after 30 seconds, or when the Stop action is called.
- `StartUpLightDimmer`: Use this and the StartDownLightDimmer instead of StartDimmer if your device has two dimmable lights.
- `StartDownLightDimmer`: The counterpart to StartUpLightDimmer