
Plum Garage Controller Documentation

Release 0

contributors

April 10, 2013

CONTENTS

1	Design Document	3
1.1	Terms and Broad Ideas	3
1.2	Realization	3
2	Indices and tables	5

Contents:

DESIGN DOCUMENT

1.1 Terms and Broad Ideas

1.1.1 Devices

A device is anything which is attached to the Plum Garage Controller system. Devices can be physical, like a switch or a sensor, or they can be software, like a program which measures the position of the sun through the day and triggers events like “sundown” or “dawn”. Devices have some combination of actions and events. A device can have both, and must define one or more actions or events.

A device can keep some information in the system, like last state.

1.1.2 Events

Events are something that the controller detects or becomes aware of. A device can trigger events like the state of a switch changing or a new device joining the system.

1.1.3 Actions

Actions are things that the controller causes.

1.1.4 Recipes

A recipe is a bit of code written to interact with a certain event, and then to do something interesting. A recipe can trigger one or more new actions or events.

1.2 Realization

1.2.1 Event Processing

When an even is triggered, it will be put onto a queue to be processed later. Processing will search for all recipes which are registered to the event. It will then add each recipe to the queue to be processed individually.

1.2.2 Device Capabilities

To write a device driver, you can inherit a base device class. This class will provide some persistence mechanisms to persist whatever information (in a key->value format) needed to manage state.

Events are realized as a key, which should include the device type, and event type, and may include things like the a device identifier; and a list of zero or more arguments which will be provided to attached recipes.

A convenience method is also provided for triggering events.

All Events and actions should be defined in the device module's own documentation.

The current implementation of device events relies on celery, with redis as a queue and a database.

The current implementation of device persistence uses json stored on the filesystem as a document store. It does not provide Consistency, Availability, or Partition Tolerance.

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*