

---

# **Consular Documentation**

***Release 1.3.0***

**Simon de Haan**

August 23, 2016



<b>1</b>	<b>Usage</b>	<b>3</b>
<b>2</b>	<b>Installing for local dev</b>	<b>5</b>
<b>3</b>	<b>Running tests</b>	<b>7</b>
<b>4</b>	<b>Consular Class Documentation</b>	<b>9</b>
	<b>Python Module Index</b>	<b>13</b>



Consular is a micro-service that relays information between [Marathon](#) and [Consul](#). It registers itself for HTTP event callbacks with [Marathon](#) and makes the appropriate API calls to register applications that [Marathon](#) runs as services in [Consul](#). De-registrations of applications happens in the same way.

[Marathon](#) is always considered the source of truth.

If Marathon application definitions contain [labels](#) (application metadata) they will be added to the [Consul](#) key/value store. This can be especially useful when [Consul](#) is combined with [Consul Template](#) to automatically generate configuration files for proxies such as [HAProxy](#) and [Nginx](#).



---

## Usage

---

```
$ pip install consular  
$ consular --help
```





---

### Installing for local dev

---

```
$ git clone https://github.com/universalcore/consular.git
$ cd consular
$ virtualenv ve
$ source ve/bin/activate
(ve)$ pip install -e .
(ve)$ pip install -r requirements-dev.txt
```



---

## Running tests

---

```
$ source ve/bin/activate  
(ve)$ py.test consular
```



---

## Consular Class Documentation

---

**class** `consular.main.Consular` (*consul\_endpoint*, *marathon\_endpoint*, *enable\_fallback*, *registration\_id*)

### Parameters

- **consul\_endpoint** (*str*) – The HTTP endpoint for Consul (often <http://example.org:8500>).
- **marathon\_endpoint** (*str*) – The HTTP endpoint for Marathon (often <http://example.org:8080>).
- **enable\_fallback** (*bool*) – Fallback to the main Consul endpoint for registrations if unable to reach Consul running on the machine running a specific Marathon task.
- **registration\_id** (*str*) – A unique parameter for this Consul server. It is used for house-keeping purposes such as purging tasks that are no longer running in Marathon.

**app\_id\_tag** (*app\_id*)

Get the app ID tag for the given app ID.

**check\_apps\_namespace\_clash** (*apps*)

Checks if app names in Marathon will cause a namespace clash in Consul. Throws an exception if there is a collision, else returns the apps.

**Param** *apps*: The JSON list of apps from Marathon's API.

**clean\_consul\_app\_labels** (*\*args*, *\*\*kwargs*)

Delete app labels stored in the Consul k/v store under the given app name that aren't present in the given set of labels.

**deregister\_consul\_service** (*agent\_endpoint*, *service\_id*)

Deregister a service from a Consul agent.

### Parameters

- **agent\_endpoint** (*str*) – The HTTP endpoint of the Consul agent.
- **service\_id** (*str*) – The ID of the Consul service to be deregistered.

**deregister\_task\_service** (*task\_id*, *host*)

Deregister a Marathon task's service from Consul.

### Parameters

- **task\_id** (*str*) – The ID of the task, this will be used as the Consul service ID.
- **host** (*str*) – The host address of the machine the task is running on.

**events** (*request*)

Listens to incoming events from Marathon on `/events`.

**Parameters** **request** (*klein.app.KleinRequest*) – The Klein HTTP request

**get\_app\_id\_from\_tags** (*tags*)

Get the app ID from the app ID tag in the given tags, or None if the tag could not be found.

**get\_consul\_app\_keys** (*app\_name*)

Get the Consul k/v keys for the app with the given name.

**get\_consul\_consular\_keys** ()

Get the next level of Consul k/v keys at ‘consular/’, i.e. will return ‘consular/my-app’ but not ‘consular/my-app/my-label’.

**handle\_status\_update\_event** (*request, event*)

Handles status updates from Marathon.

The various task stages are handled as follows:

TASK\_STAGING: ignored TASK\_STARTING: ignored TASK\_RUNNING: task data updated on Consul TASK\_FINISHED: task data removed from Consul TASK\_FAILED: task data removed from Consul TASK\_KILLED: task data removed from Consul TASK\_LOST: task data removed from Consul

**Parameters**

- **request** (*klein.app.KleinRequest*) – The Klein HTTP request
- **event** (*dict*) – The Marathon event

**purge\_dead\_app\_labels** (*\*args, \*\*kwargs*)

Delete any keys stored in the Consul k/v store that belong to apps that no longer exist.

**Param** **apps**: The list of apps as returned by the Marathon API.

**put\_consul\_app\_labels** (*app\_name, labels*)

Store the given set of labels under the given app name in the Consul k/v store.

**put\_consul\_kvs** (*\*args, \*\*kwargs*)

Store the given key/value set in the Consul k/v store.

**reg\_id\_tag** ()

Get the registration ID tag for this instance of Consular.

**register\_marathon\_event\_callback** (*\*args, \*\*kwargs*)

Register Consular with Marathon to receive HTTP event callbacks. To use this ensure that [Marathon is configured](#) to send HTTP event callbacks for state changes in tasks.

**Parameters** **events\_url** (*str*) – The HTTP endpoint to register with Marathon for event callbacks.

**register\_task\_service** (*app\_id, task\_id, host, ports*)

Register a Marathon task as a service in Consul.

**Parameters**

- **app\_id** (*str*) – The ID of the Marathon app that the task belongs to.
- **task\_id** (*str*) – The ID of the task, this will be used as the Consul service ID.
- **host** (*str*) – The host address of the machine the task is running on.
- **ports** (*list*) – The port numbers the task can be accessed on on the host machine.

**run** (*host, port*)

Starts the HTTP server.

### Parameters

- **host** (*str*) – The host to bind to (example is `localhost`)
- **port** (*int*) – The port to listen on (example is `7000`)

**schedule\_sync** (*interval*, *purge=False*)

Schedule a recurring sync of apps, starting after this method is called.

### Parameters

- **interval** (*float*) – The number of seconds between syncs.
- **purge** (*bool*) – Whether to purge old apps after each sync.

**Returns** A tuple of the `LoopingCall` object and the deferred created when it was started.

**sync\_app\_labels** (*\*args*, *\*\*kwargs*)

Sync the app labels for the given app by pushing its labels to the Consul k/v store and cleaning any labels there that are no longer present.

**Param** app: The app JSON as return by the Marathon HTTP API.

**sync\_apps** (*\*args*, *\*\*kwargs*)

Ensure all the apps in Marathon are registered as services in Consul.

Set `purge` to `True` if you automatically want services in Consul that aren't registered in Marathon to be purged. Consular only purges services that have been registered with the same `registration-id`.

**Parameters** **purge** (*bool*) – To purge or not to purge.

**update\_task\_running** (*\*args*, *\*\*kwargs*)

Use a running event to register a new Consul service.

`consular.main.get_app_name` (*app\_id*)

Get the app name from the marathon app ID. Separators in the ID (`/`) are replaced with `-`'s while the leading separator is removed.

`consular.main.handle_not_found_error` (*\*args*, *\*\*kwargs*)

Perform a request and catch the not found (404) error if one occurs.

**Param** f: The function to call to perform the request. The function may return a deferred.

**Param** args: The arguments to call the function with.

**Param** kwargs: The keyword arguments to call the function with.

**Returns** The return value of the function call or `None` if there was a 404 response code.





**C**

`consular.main`, 9



## A

`app_id_tag()` (consular.main.Consular method), 9

## C

`check_apps_namespace_clash()` (consular.main.Consular method), 9

`clean_consul_app_labels()` (consular.main.Consular method), 9

`Consular` (class in consular.main), 9

`consular.main` (module), 9

## D

`deregister_consul_service()` (consular.main.Consular method), 9

`deregister_task_service()` (consular.main.Consular method), 9

## E

`events()` (consular.main.Consular method), 9

## G

`get_app_id_from_tags()` (consular.main.Consular method), 10

`get_app_name()` (in module consular.main), 11

`get_consul_app_keys()` (consular.main.Consular method), 10

`get_consul_consular_keys()` (consular.main.Consular method), 10

## H

`handle_not_found_error()` (in module consular.main), 11

`handle_status_update_event()` (consular.main.Consular method), 10

## P

`purge_dead_app_labels()` (consular.main.Consular method), 10

`put_consul_app_labels()` (consular.main.Consular method), 10

`put_consul_kvs()` (consular.main.Consular method), 10

## R

`reg_id_tag()` (consular.main.Consular method), 10

`register_marathon_event_callback()` (consular.main.Consular method), 10

`register_task_service()` (consular.main.Consular method), 10

`run()` (consular.main.Consular method), 10

## S

`schedule_sync()` (consular.main.Consular method), 11

`sync_app_labels()` (consular.main.Consular method), 11

`sync_apps()` (consular.main.Consular method), 11

## U

`update_task_running()` (consular.main.Consular method), 11