# Cyclotron Consul Documentation

***Release 0.1.0***

**R. Picard**

**Jun 08, 2020**

# CONTENTS:


**1 Get Started** **1**

1.1 KV Store . . . . . 1

**2 Reference** **5**

**3 Indices and tables** **7**

**Python Module Index** **9**

**Index** **11**

CHAPTER

ONE

# GET STARTED

This packages exposes one adapter per consul API family.

## 1.1 KV Store

One can Access the KV store with the kv adapter:

The constructor of the kv adapter takes an observable of http_response as input, and returns an Adapter object. The Adapter object contains two properties:

- sink: An observable of http requests (requests sent to the consul endpoint)
- api: An accessor to the KV Store APIs

### 1.1.1 Read the value of a key in the KV Store

Reading a key is done the following way:

```
import cyclotron_consul.kv as kv

kv_adapter = kv.adapter(source.http.response)

value = kv_adapter.api.read_key("http://localhost:8500", "test")
```

Where value is an observable that emits a single item with a Key/Value object.

Here is a full example:

```
from collections import namedtuple

import rx
import rx.operators as ops

from cyclotron import Component
from cyclotron.asyncio.runner import run
import cyclotron_aiohttp.http as http
import cyclotron_std.sys.stdout as stdout

import cyclotron_consul.kv as kv

ReadKeySource = namedtuple('ReadKeySource', ['http'])
ReadKeySink = namedtuple('ReadKeySink', ['http', 'stdout'])
ReadKeyDrivers = namedtuple('ReadKeyDrivers', ['http', 'stdout'])
```

(continues on next page)

(continued from previous page)

```
def read_key(source):
    kv_adapter = kv.adapter(source.http.response)

    value = kv_adapter.api.read_key("http://localhost:8500", "test").pipe(
        ops.map(lambda i: "key: {}, value: {}".format(i.key, i.value))
    )

    return ReadKeySink(
        http=http.Sink(request=kv_adapter.sink),
        stdout=stdout.Sink(data=value),
    )


def main():
    run(Component(call=read_key, input=ReadKeySource),
        ReadKeyDrivers(
            http=http.make_driver(),
            stdout=stdout.make_driver(),
        )
    )


if __name__ == '__main__':
    main()
```

### 1.1.2 Watch the value of a key in the KV Store

This is similar to reading a key, except that the result observable emits a new items each time the value of the key changes:

```
import cyclotron_consul.kv as kv

kv_adapter = kv.adapter(source.http.response)

value = kv_adapter.api.watch_key("http://localhost:8500", "test")
```

Where value is an observable that emits an item with a Key/Value object each time the value of the key is updated on consul.

Here is a full example:

```
from collections import namedtuple

import rx
import rx.operators as ops

from cyclotron import Component
from cyclotron.asyncio.runner import run
import cyclotron_aiohttp.http as http
import cyclotron_std.sys.stdout as stdout

import cyclotron_consul.kv as kv
```

(continues on next page)

(continued from previous page)

```
ReadKeySource = namedtuple('ReadKeySource', ['http'])
ReadKeySink = namedtuple('ReadKeySink', ['http', 'stdout'])
ReadKeyDrivers = namedtuple('ReadKeyDrivers', ['http', 'stdout'])


def read_key(source):
    kv_adapter = kv.adapter(source.http.response)

    value = kv_adapter.api.watch_key("http://localhost:8500", "test").pipe(
        ops.map(lambda i: "key: {}, value: {}".format(i.key, i.value)),
    )

    return ReadKeySink(
        http=http.Sink(request=kv_adapter.sink),
        stdout=stdout.Sink(data=value),
    )


def main():
    run(Component(call=read_key, input=ReadKeySource),
        ReadKeyDrivers(
            http=http.make_driver(),
            stdout=stdout.make_driver(),
        )
    )


if __name__ == '__main__':
    main()
```

CHAPTER

TWO

# REFERENCE

**class** cyclotron_consul.kv.**Adapter**(*sink*, *api*)

**property api**
Alias for field number 1

**property sink**
Alias for field number 0

**class** cyclotron_consul.kv.**Api**(*read_key*, *watch_key*)

**property read_key**
Alias for field number 0

**property watch_key**
Alias for field number 1

**class** cyclotron_consul.kv.**KeyValue**(*key*, *value*)

**property key**
Alias for field number 0

**property value**
Alias for field number 1

cyclotron_consul.kv.**adapter**(*source*)
Creates a consul adapter for the KV API.

**Parameters** **source** – an aiohttp response stream.

**Returns** A Client object

cyclotron_consul.kv.**read_key**(*http_client*, *endpoint*, *key*)
Reads a key on the specified endpoint

**Parameters**

- **http_client** – An instance of http client. This parameter is already binded when called from the adapter.
- **endpoint** – the consul server full url
- **key** – The key to read

**Returns** On observable that emits a single KeyValue item if the request succeeds. Otherwise the observable completes on error.

### Example

>>>import cyclotron_consul.kv as kv >>>client = kv.client(http_source) >>>client.api.read_key("http://localhost:8500", "mykey").subscribe( >>> on_next=print >>>) >>># forward client.sink to http driver sink

`cyclotron_consul.kv.`**`watch_key`**(*http_client*, *endpoint*, *key*)

Reads a key on the specified endpoint, and watch for updates.

**Parameters**

- **`http_client`** – An instance of http client. This parameter is already binded when called from the adapter.
- **`endpoint`** – the consul server full url
- **`key`** – The key to read

**Returns** On observable that emits a KeyValue item each time the value associated to the key is updated on consul. In case of error, the observable completes on error.

### Example

>>>import cyclotron_consul.kv as kv >>>client = kv.client(http_source) >>>client.api.watch_key("http://localhost:8500", "mykey").subscribe( >>> on_next=print >>>) >>># forward client.sink to http driver sink

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## C

cyclotron_consul.kv, 5

# INDEX


## A

`Adapter` (*class in cyclotron_consul.kv*), 5
`adapter()` (*in module cyclotron_consul.kv*), 5
`Api` (*class in cyclotron_consul.kv*), 5
`api()` (*cyclotron_consul.kv.Adapter property*), 5

## C

`cyclotron_consul.kv`
    `module`, 5

## K

`key()` (*cyclotron_consul.kv.KeyValue property*), 5
`KeyValue` (*class in cyclotron_consul.kv*), 5

## M

`module`
    `cyclotron_consul.kv`, 5

## R

`read_key()` (*cyclotron_consul.kv.Api property*), 5
`read_key()` (*in module cyclotron_consul.kv*), 5

## S

`sink()` (*cyclotron_consul.kv.Adapter property*), 5

## V

`value()` (*cyclotron_consul.kv.KeyValue property*), 5

## W

`watch_key()` (*cyclotron_consul.kv.Api property*), 5
`watch_key()` (*in module cyclotron_consul.kv*), 6