kismet_rest Documentation

Release 2019.05.02

Mike Kershaw / Dragorn

May 25, 2019

Contents

Python wrapper for Kismet REST interface.

CHAPTER 1

Installing from PyPI

```
pip install kismet_rest
```

CHAPTER 2

Installing from source

```
git clone https://github.com/kismetwireless/python-kismet-rest
cd python-kismet-rest && pip install .
```

CHAPTER 3

Usage examples

3.1 Legacy functionality (KismetConnector):

```
import kismet_rest
conn = kismet_rest.KismetConnector(username="my_user", password="my_pass")
for device in conn.device_summary():
    pprint.pprint(device)
```

3.2 Alerts since 2019-01-01:

```
import kismet_rest
alerts = kismet_rest.Alerts()
for alert in alerts.all(ts_sec=1546300800):
    print(alert)
```

3.3 Devices last observed since 2019-01-01:

```
import kismet_rest
devices = kismet_rest.Devices()
for device in devices.all(ts=1546300800):
    print(device)
```

CHAPTER 4

Developer notes:

- Formatting commit messages: * Correctly-formatted commit messages will be organized in CHANGELOG.rst * Commit messages are formatted like this `type: audience: message !tag` * Type is for the type of change (`new`, `chg`) * Audience is for the audience of the commit note(`usr`,‘‘test‘‘,‘‘doc‘‘) * The message part is pretty self-explanatory. * The optional tag allows you to flag a commit for exclusion from CHANGELOG.rst.(`minor` or `wip`) * A commit message like this: `new: usr: Made a new widget.` will appear in CHANGELOG.rst, under the appropriate release, under the “New” section. * More info on message formatting: https://github.com/vaab/gitchangelog
- Updating CHANGELOG.rst: * Install gitchangelog: `pip3 install gitchangelog` * Make sure that `__version__` is correct in `kismet_rest/__init__.py` * Build the new changelog: `gitchangelog > CHANGELOG.rst`

CHAPTER 5

Table of Contents

5.1 Abstractions

This library presents endpoints as Python objects.

5.1.1 Alerts

class kismet_rest.**Alerts**(*host_uri='http://127.0.0.1:2501'*, *session-cache_path='~/.pykismet_session'*, ***kwargs*)

Alerts abstraction.

all(*callback=None*, *callback_args=None*, ***kwargs*)

Yield all alerts, one at a time.

If callback is set, nothing will be returned.

Parameters

- **callback** – Callback function.
- **callback_args** – Arguments for callback.

Keyword Arguments

- **ts_sec** (*int*) – Starting timestamp in seconds since Epoch.
- **ts_usec** (*int*) – Microseconds for starting timestamp.

Yields *dict* – Alert json, or None if callback is set.

define(*name*, *description*, *rate='10/min'*, *burst='1/sec'*, *phyname=None*)

Define an alert.

LOGIN REQUIRED

Define a new alert. This alert can then be triggered on external conditions via raise_alert(...)

Phyname is optional, and links the alert to a specific PHY type.

Rate and Burst are optional rate and burst limits.

Parameters

- **name** (*str*) – Name of alert.
- **description** (*str*) – Description of alert.
- **rate** (*str*) – Rate limit. Defaults to `10/min`.
- **burst** (*str*) – Burst limit. Defaults to `1/sec`.
- **phyname** (*str*) – Name of PHY. Defaults to None.

Returns True for success, False for failed request.

Return type bool

raise_alert(*name*, *text*, *bssid=None*, *source=None*, *dest=None*, *other=None*, *channel=None*)
Raise an alert in Kismet.

Trigger an alert; the alert can be one defined via define_alert(...) or an alert built into the system.

The alert name and content of the alert are required, all other fields are optional.

Parameters

- **name** (*str*) – Name of alert.
- **text** (*str*) – Descriptive text for alert.
- **bssid** (*str*) – BSSID to filter for.
- **source** (*str*) – ...
- **dest** (*str*) – ...
- **other** (*str*) – ...
- **channel** (*str*) – Channel to filter for.

5.1.2 Datasources

class kismet_rest.**Datasources**(*host_uri='http://127.0.0.1:2501'*, *session-cache_path='~/.pykismet_session'*, ***kwargs*)
Datasources abstraction.

add(*source*)
Add a new source to Kismet.

source is a standard source definition.

Requires valid login.

Returns Success

Return type bool

all(*callback=None*, *callback_args=None*)
Yield all datasources, one at a time.

If callback is set, nothing will be returned.

Parameters

- **callback** – Callback function.
- **callback_args** – Arguments for callback.

Yields *dict* – Datasource json, or None if callback is set.

interfaces(*callback=None*, *callback_args=None*)
Yield all interfaces, one at a time.

If callback is set, nothing will be returned.

Parameters

- **callback** – Callback function.
- **callback_args** – Arguments for callback.

Yields *dict* – Datasource json, or None if callback is set.

pause(*source*)
Pause source.

Parameters **source** (*str*) – UUID of source to pause.

Returns Success

Return type bool

resume(*source*)
Resume paused source.

Parameters **source** (*str*) – UUID of source to resume.

Returns Success

Return type bool

set_channel(*uuid*, *channel*)
Return `True` if operation was successful, `False` otherwise.

Locks an data source to an 802.11 channel or frequency. Channel may be complex channel such as "6HT40+".

Requires valid login.

set_hop(*uuid*)
Configure a source for hopping.

Uses existing source hop / channel list / etc attributes.

Requires valid login

set_hop_channels(*uuid*, *rate*, *channels*)
Set datasource hopping rate by UUID.

Configures a data source for hopping at 'rate' over a vector of channels.

Requires valid login

set_hop_rate(*uuid*, *rate*)
Set the hop rate of a specific data source by UUID.

Configures the hopping rate of a data source, while not changing the channels used for hopping.

Requires valid login

5.1.3 Devices

class kismet_rest.**Devices**(*host_uri='http://127.0.0.1:2501'*, *session-cache_path='~/.pykismet_session'*, ***kwargs*)

Devices abstraction.

all(*callback=None*, *callback_args=None*, ***kwargs*)

Yield all devices, one at a time.

If callback is set, nothing will be returned.

Parameters

- **callback** – Callback function.
- **callback_args** – Arguments for callback.

Keyword Arguments **ts** (*int*) – Starting last-seen timestamp in seconds since Epoch.

Yields *dict* – Device json, or None if callback is set.

by_key(*device_key*, *field=None*, *fields=None*)

Return a dictionary representing one device, identified by `key`.

Fetch a complete device record by the Kismet key (unique key per Kismet session) or fetch a specific sub-field by path.

Returns Dictionary object describing one device.

Return type dict

by_mac(*callback=None*, *callback_args=None*, ***kwargs*)

Yield devices matching provided MAC addresses or masked MAC groups.

Parameters

- **callback** – Callback function.
- **callback_args** – Arguments for callback.

Keyword Arguments

- **devices** (*list*) – List of device MACs or MAC masks.
- **fields** (*list*) – List of fields to return.

Yields *dict* – Device json, or None if callback is set.

dot11_access_points(*callback=None*, *callback_args=None*, ***kwargs*)

Return a list of dot11 access points.

List devices which are considered to be 802.11 access points, using the /devices/views/phydot11_accesspoints/ view

Returned devices can be summarized/simplified by the fields list.

If a timestamp is given, only devices modified more recently than the timestamp (and matching any other conditions) will be returned.

If a regex is given, only devices matching the regex (and any other conditions) will be returned.

If a callback is given, it will be called for each device in the result. If no callback is provided, the results will be yielded as dictionary objects.

Parameters

- **callback** (*obj*) – Callback for processing individual results.

- **cbargs** (*list*) – List of arguments for callback.

Keyword Arguments

- **last_time** (*int*) – Unix epoch timestamp
- **regex** (*str*) – Regular expression for filtering results.
- **fields** (*list*) – Fields for filtering.

Yields *dict* –

Dictionary-type objects which describe access points. Keys describing access points: `dot11.device`, `kismet.device.base.basic_crypt_set`, `kismet.device.base.basic_type_set`, `kismet.device.base.channel`, `kismet.device.base.commonname`, `kismet.device.base.crypt`, `kismet.device.base.datasize`, `kismet.device.base.datasize.rrd`, `kismet.device.base.first_time`, `kismet.device.base.freq_khz_map`, `kismet.device.base.frequency`, `kismet.device.base.key`, `kismet.device.base.last_time`, `kismet.device.base.macaddr`, `kismet.device.base.manuf`, `kismet.device.base.mod_time`, `kismet.device.base.name`, `kismet.device.base.num_alerts`, `kismet.device.base.packet.bin.250`, `kismet.device.base.packet.bin.500`, `kismet.device.base.packets.crypt`, `kismet.device.base.packets.data`, `kismet.device.base.packets.error`, `kismet.device.base.packets.filtered`, `kismet.device.base.packets.llc`, `kismet.device.base.packets.rrd`, `kismet.device.base.packets.rx`, `kismet.device.base.packets.total`, `kismet.device.base.packets.tx`, `kismet.device.base.phyname`, `kismet.device.base.seenby`, `kismet.device.base.server_uuid`, `kismet.device.base.signal`, `kismet.device.base.tags`, `kismet.device.base.type`.

dot11_clients_of(*ap_id*, *callback=None*, *callback_args=None*, ***kwargs*)

List clients of an 802.11 AP.

List devices which are clients of a given 802.11 access point, using the /phy/phy80211/clients-of endpoint.

Returned devices can be summarized/simplified by the fields list.

If a callback is given, it will be called for each device in the result. If no callback is provided, the results will be yielded.

Parameters

- **ap_id** (*str*) – ID of AP to return clients for.
- **callback** – Callback function.
- **callback_args** – Arguments for callback.

Yields *dict* – Dictionary describing a client of the identified AP.

5.1.4 GPS

class kismet_rest.**GPS**(*host_uri='http://127.0.0.1:2501'*, *sessioncache_path='~/.pykismet_session'*, ***kwargs*)

GPS abstraction.

current_location()

Return the gps location.

Returns

Dictionary object describing current location of Kismet server. Keys represented in output: `kismet.common.location.lat`, `kismet.common.location.lon`, `kismet.common.location.alt`, `kismet.common.location.heading`, `kismet.common.location.speed`, `kismet.common.location.time_sec`, `kismet.common.location.time_usec`, `kismet.common.location.fix`, `kismet.common.location.valid`

Return type dict

5.1.5 Messages

class kismet_rest.**Messages**(*host_uri='http://127.0.0.1:2501'*, *session-cache_path='~/.pykismet_session'*, ***kwargs*)

Messages abstraction.

all(*callback=None*, *callback_args=None*, ***kwargs*)

Yield all messages, one at a time.

If callback is set, nothing will be returned.

Parameters

- **callback** – Callback function.
- **callback_args** – Arguments for callback.

Keyword Arguments

- **ts_sec** (*int*) – Seconds since epoch for first message retrieved.
- **ts_usec** (*int*) – Microseconds modifier for ts_sec query argument.

Yields *dict* – Message json, or None if callback is set.

5.1.6 System

class kismet_rest.**System**(*host_uri='http://127.0.0.1:2501'*, *session-cache_path='~/.pykismet_session'*, ***kwargs*)

Wrap all interaction with /system/ endpoint.

get_status()

Return json representing Kismet system status.

get_system_time(*time_format=None*)

Return current time from Kismet REST API.

Parameters **format** (*str or None*) – Format time before returning. Supported formats: None (return as dict), `iso` (ISO 8601). Defaults to None.

5.2 Legacy

class kismet_rest.**KismetConnector**(*host_uri='http://127.0.0.1:2501'*, *session-cache_path='~/.pykismet_session'*, ***kwargs*)

Kismet rest API.

add_datasource(*source*)
Add a new source to Kismet.

Deprecated. Use *kismet_rest.Datasources.add()* instead.

source is a standard source definition.

Requires valid login.

Returns Success

Return type bool

alerts(*ts_sec=0*, *ts_usec=0*)
Return alert object.

Deprecated. Use *kismet_rest.Alerts.all()* instead.

Fetch alert object, containing metadata and list of alerts, optionally filtered to alerts since a given timestamp

Parameters

- **ts_sec** (*int*) – Timestamp seconds (Unix epoch)
- **ts_usec** (*int*) – Timestamp microseconds

Returns

Dictionary containing metadata and a list of alerts. Keys represented in output: `'kismet.alert.timestamp`, `kismet.alert.list`.

Return type dict

config_datasource_set_channel(*uuid*, *channel*)
Return `True` if operation was successful, `False` otherwise.

Deprecated. Use *kismet_rest.Datasources.set_channel()* instead.

Locks an data source to an 802.11 channel or frequency. Channel may be complex channel such as "6HT40+".

Requires valid login.

config_datasource_set_hop(*uuid*)
Configure a source for hopping.

Deprecated. Use *kismet_rest.Datasources.set_hop()* instead.

Uses existing source hop / channel list / etc attributes.

Requires valid login

config_datasource_set_hop_channels(*uuid*, *rate*, *channels*)
Set datasource hopping rate by UUID.

Deprecated. Use *kismet_rest.Datasources.set_hop_channels()* instead.

Configures a data source for hopping at 'rate' over a vector of channels.

Requires valid login

config_datasource_set_hop_rate(*uuid*, *rate*)
Set the hop rate of a specific data source by UUID.

Deprecated. Use *kismet_rest.Datasources.set_hop_rate()* instead.

Configures the hopping rate of a data source, while not changing the channels used for hopping.

Requires valid login

datasource_list_interfaces()
Return a list of all available interfaces.

Deprecated. Use `kismet_rest.Datasources.yield_interfaces()` instead.

datasources()
Return a list of data sources.

Deprecated. Use *kismet_rest.Datasources.all()* instead.

Returns

List of dictionary-type objects, which describe data sources. Dictionary keys are: `kismet.datasource.capture_interface`, `kismet.datasource.channel`, `kismet.datasource.channels`, `kismet.datasource.definition`, `kismet.datasource.dlt`, `kismet.datasource.error`, `kismet.datasource.error_reason`, `kismet.datasource.hardware`, `kismet.datasource.hop_channels`, `kismet.datasource.hop_offset`, `kismet.datasource.hopping`, `kismet.datasource.hop_rate`, `kismet.datasource.hop_shuffle`, `kismet.datasource.hop_shuffle_skip`, `kismet.datasource.hop_split`, `kismet.datasource.info.amp_gain`, `kismet.datasource.info.amp_type`, `kismet.datasource.info.antenna_beamwidth`, `kismet.datasource.info.antenna_gain`, `kismet.datasource.info.antenna_orientation`, `kismet.datasource.info.antenna_type`, `kismet.datasource.interface`, `kismet.datasource.ipc_binary`, `kismet.datasource.ipc_pid`, `kismet.datasource.linktype_override`, `kismet.datasource.name`, `kismet.datasource.num_error_packets`, `kismet.datasource.num_packets`, `kismet.datasource.packets_rrd`, `kismet.datasource.passive`, `kismet.datasource.paused`, `kismet.datasource.remote`, `kismet.datasource.retry`, `kismet.datasource.retry_attempts`, `kismet.datasource.running`, `kismet.datasource.source_key`, `kismet.datasource.source_number`, `kismet.datasource.total_retry_attempts`, `kismet.datasource.type_driver`, `kismet.datasource.uuid`, `kismet.datasource.warning`.

Return type list

define_alert(*name*, *description*, *rate*, *burst*) → Boolean
Deprecated. Use *kismet_rest.Alerts.define()* instead.

LOGIN REQUIRED

Define a new alert. This alert can then be triggered on external conditions via raise_alert(...)

Phyname is optional, and links the alert to a specific PHY type.

Rate and Burst are optional rate and burst limits.

device(*key*, *field=None*, *fields=None*)
Wrap device_by_key.

Deprecated.

device_by_key(*key*, *field=None*, *fields=None*)
Return a dictionary representing one device, identified by `key`.

Note: This is superseded by `kismet_rest.Devices.get_by_key()`

Fetch a complete device record by the Kismet key (unique key per Kismet session) or fetch a specific sub-field by path.

If a field simplification set is passed in 'fields', perform a simplification on the result

device_by_mac (*mac*, *fields=None*)
Return a list of all devices matching `mac`.

Deprecated. Use `kismet_rest.Devices.yield_by_mac()` instead.

Return a vector of all devices in all phy types matching the supplied MAC address; typically this will return a vector of a single device, but MAC addresses could overlap between phy types.

If a field simplification set is passed in 'fields', perform a simplification on the result

device_field (*key*, *field*)
Wrap device_by_key.

Deprecated, prefer device_by_key with field.

device_list (*callback=None*, *cbargs=None*)
Return all fields of all devices.

Note: This is superseded by *kismet_rest.Devices.all()*

This may be extremely memory and CPU intensive and should be avoided. Memory use can be reduced by providing a callback, which will be invoked for each device.

In general THIS API SHOULD BE AVOIDED. There are several potentially serious repercussions in querying all fields of all devices in a very high device count environment.

It is strongly recommended that you use smart_device_list(...)

device_list_by_mac (*maclist*, *fields=None*, *callback=None*, *cbargs=None*)
List devices matching MAC addresses in maclist.

Note: This method is deprecated.

Use `kismet_rest.Devices.yield_by_mac()` instead.

MAC addresses may be complete MACs or masked MAC groups ("AA:BB:CC:00:00:00/FF:FF:FF:00:00:00").

Returned devices can be summarized/simplified by the fields list.

If a callback is given, it will be called for each device in the result. If no callback is provided, the results will be returned as a vector.

device_summary (*callback=None*, *cbargs=None*)
Return a summary of all devices.

Note: This is superseded by *kismet_rest.Devices.all()*

Deprecated API - now referenced as device_list(..)

device_summary_since (*ts*[, *fields*, *callback*, *cbargs*])
device summary list

Note: This is superseded by *kismet_rest.Devices.all()*

Deprecated API - now referenced as smart_device_list(...)

Return object containing summary of devices added or changed since ts and ts info

dot11_access_points (*tstamp=None*, *regex=None*, *fields=None*, *callback=None*, *cbargs=None*)
Return a list of dot11 access points.

Note: This is superseded by *kismet_rest.Devices.dot11_access_points()*

List devices which are considered to be 802.11 access points, using the /devices/views/phydot11_accesspoints/ view

Returned devices can be summarized/simplified by the fields list.

If a timestamp is given, only devices modified more recently than the timestamp (and matching any other conditions) will be returned.

If a regex is given, only devices matching the regex (and any other conditions) will be returned.

If a callback is given, it will be called for each device in the result. If no callback is provided, the results will be returned as a vector.

Parameters

- **ts** (*int*) – Unix epoch timestamp
- **regex** (*str*) – Regular expression for filtering results.
- **fields** (*list*) – Fields for filtering.
- **callback** (*obj*) – Callback for processing individual results.
- **cbargs** (*list*) – List of arguments for callback.

Returns

List of dictionary-type objects which describe access points. Keys describing access points: `dot11.device`, `kismet.device.base.basic_crypt_set`, `kismet.device.base.basic_type_set`, `kismet.device.base.channel`, `kismet.device.base.commonname`, `kismet.device.base.crypt`, `kismet.device.base.datasize`, `kismet.device.base.datasize.rrd`, `kismet.device.base.first_time`, `kismet.device.base.freq_khz_map`, `kismet.device.base.frequency`, `kismet.device.base.key`, `kismet.device.base.last_time`, `kismet.device.base.macaddr`, `kismet.device.base.manuf`, `kismet.device.base.mod_time`, `kismet.device.base.name`, `kismet.device.base.num_alerts`, `kismet.device.base.packet.bin.250`, `kismet.device.base.packet.bin.500`, `kismet.device.base.packets.crypt`, `kismet.device.base.packets.data`, `kismet.device.base.packets.error`, `kismet.device.base.packets.filtered`, `kismet.device.base.packets.llc`, `kismet.device.base.packets.rrd`, `kismet.device.base.packets.rx`, `kismet.device.base.packets.total`, `kismet.device.base.packets.tx`, `kismet.device.base.phyname`, `kismet.device.base.seenby`, `kismet.device.base.server_uuid`, `kismet.device.base.signal`, `kismet.device.base.tags`, `kismet.device.base.type`.

Return type list

dot11_clients_of(*apkey*, *fields=None*, *callback=None*, *cbargs=None*)

List clients of 802.11 AP.

Note: This is superseded by *kismet_rest.Devices.dot11_clients_of()*

List devices which are clients of a given 802.11 access point, using the /phy/phy80211/clients-of endpoint.

Returned devices can be summarized/simplified by the fields list.

If a callback is given, it will be called for each device in the result. If no callback is provided, the results will be returned as a vector.

location()
Return the gps location.

Deprecated. Use *kismet_rest.GPS.current_location()* instead.

Returns

Dictionary object describing current location of Kismet server. Keys represented in output: `kismet.common.location.lat`, `kismet.common.location.lon`, `kismet.common.location.alt`, `kismet.common.location.heading`, `kismet.common.location.speed`, `kismet.common.location.time_sec`, `kismet.common.location.time_usec`, `kismet.common.location.fix`, `kismet.common.location.valid`

Return type dict

messages(*ts_sec=0*, *ts_usec=0*)
Return message object.

Deprecated. Use *kismet_rest.Messages.all()* instead.

Fetch message object, containing metadata and list of messages, optionally filtered to messages since a given timestamp

Parameters

- **ts_sec** (*int*) – Timestamp seconds (Unix epoch)
- **ts_usec** (*int*) – Timestamp microseconds

Returns

Dictionary containing metadata and a list of messages. Top-level keys: `kismet.messagebus.timestamp`, `kismet.messagebus.list`

Return type dict

raise_alert(*name*, *text*, *bssid=None*, *source=None*, *dest=None*, *other=None*, *channel=None*)
Raise an alert in Kismet.

Deprecated. Use `kismet_rest.Alerts.raise()` instead.

LOGIN REQUIRED

Trigger an alert; the alert can be one defined via define_alert(...) or an alert built into the system.

The alert name and content of the alert are required, all other fields are optional.

Parameters

- **name** (*str*) – Name of alert.
- **text** (*str*) – Descriptive text for alert.
- **bssid** (*str*) – BSSID to filter for.
- **source** (*str*) – ...
- **dest** (*str*) – ...
- **other** (*str*) – ...
- **channel** (*str*) – Channel to filter for.

smart_device_list(*ts=0*, *fields=None*, *regex=None*, *callback=None*, *cbargs=None*)
Return a list of devices.

Note: This is superseded by *kismet_rest.Devices.all()*

Perform a 'smart' device list. The device list can be manipulated in several ways:

1. Devices active since last timestamp. By setting the 'ts' parameter, only devices which have been active since that timestamp will be returned.
2. Devices which match a regex, as defined by the regex spec above
3. Devices can be simplified to reduce the amount of work being done and number of fields being returned.

If a callback is given, it will be called for each device in the result. If no callback is provided, the results will be returned as a vector.

Parameters

- **ts** (*int*) – Unix epoch timestamp.
- **fields** (*list*) – List of field names for matching.
- **regex** (*str*) – Regular expression for field matching.
- **callback** (*obj*) – Callback for processing search results.
- **cbargs** (*list*) – List of arguments for callback.

Returns

List of dictionary-type objects, which describe devices observed by Kismet. Dictionary keys are: `dot11.device`, `kismet.device.base.basic_crypt_set`, `kismet.device.base.basic_type_set`, `kismet.device.base.channel`, `kismet.device.base.commonname`, `kismet.device.base.crypt`, `kismet.device.base.datasize`, `kismet.device.base.datasize.rrd`, `kismet.device.base.first_time`, `kismet.device.base.freq_khz_map`, `kismet.device.base.frequency`, `kismet.device.base.key`, `kismet.device.base.last_time`, `kismet.device.base.macaddr`, `kismet.device.base.manuf`, `kismet.device.base.mod_time`, `kismet.device.base.name`, `kismet.device.base.num_alerts`, `kismet.device.base.packet.bin.1000`, `kismet.device.base.packet.bin.250`, `kismet.device.base.packet.bin.500`, `kismet.device.base.packets.crypt`, `kismet.device.base.packets.data`, `kismet.device.base.packets.error`, `kismet.device.base.packets.filtered`, `kismet.device.base.packets.llc`, `kismet.device.base.packets.rrd`, `kismet.device.base.packets.rx`, `kismet.device.base.packets.total`, `kismet.device.base.packets.tx`, `kismet.device.base.phyname`, `kismet.device.base.seenby`, `kismet.device.base.server_uuid`, `kismet.device.base.signal`, `kismet.device.base.tags`, `kismet.device.base.type`.

Return type list

smart_summary_since([*ts*, *fields*, *regex*, *callback*, *cbargs*])
device summary list

Note: This is superseded by *kismet_rest.Devices.all()*

Deprecated API - now referenced as smart_device_list(...)

system_status()
Return system status.

Note: This is superseded by *kismet_rest.System.get_status()*

CHAPTER 6

Changelog

6.1 v2019.05.02

6.1.1 Changes

- Support Python 3.5. [ashmastaflash]
- Add MANIFEST.in. [ashmastaflash]

6.2 2019.05.01 (2019-05-20)

6.2.1 New

- Refactor complete. [ashmastaflash]

  Closes #1

6.2.2 Changes

- Add developer notes to README.rst. [ashmastaflash]
- Add configs for gitchangelog and rtd. [ashmastaflash]

6.2.3 Other

- Update docs. [Mike Kershaw / Dragorn]
- Start extracting module. [Mike Kershaw / Dragorn]
- Started repo. [Mike Kershaw / Dragorn]

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Index