notifiers Documentation

Release 1.2.0

Or Carmi

Jan 11, 2020

CONTENTS:

Got an app or service and you want to enable your users to use notifications with their provider of choice? Working on a script and you want to receive notification based on its output? You don't need to implement a solution yourself, or use individual provider libs. A one stop shop for all notification providers with a unified and simple interface.

Click for a list of currently supported *Providers*. See latest changes in *Changelog*.

CHAPTER

ONE

ADVANTAGES

- Spend your precious time on your own code base, instead of chasing down 3rd party provider APIs. That's what we're here for!
- With a minimal set of well known and stable dependencies (requests, jsonschema and click) you're better off than installing 3rd party SDKs.
- A unified interface means that you already support any new providers that will be added, no more work needed!
- Thorough testing means protection against any breaking API changes. We make sure your code your notifications will always get delivered!

CHAPTER

TWO

INSTALLATION

Via pip:

```
$ pip install notifiers
```

Via Dockerhub:

```
$ docker pull liiight/notifiers
```

CHAPTER

THREE

BASIC USAGE

```
>>> from notifiers import get_notifier
>>> pushover = get_notifier('pushover')
>>> pushover.required
{'required': ['user', 'message', 'token']}
>>> pushover.notify(user='foo', token='bar', message='test')
<NotificationResponse,provider=Pushover,status=Success>
```

CHAPTER

FOUR

FROM CLI

```
$ notifiers pushover notify --user foo --token baz "This is so easy!"
```

CHAPTER

FIVE

AS A LOGGER

Directly add to your existing stdlib logging:

```
>>> import logging
>>> from notifiers.logging import NotificationHandler
>>> log = logging.getLogger(__name__)
>>> defaults = {
...     'token': 'foo',
...     'user': 'bar'
... }
>>> hdlr = NotificationHandler('pushover', defaults=defaults)
>>> hdlr.setLevel(logging.ERROR)
>>> log.addHandler(hdlr)
>>> log.error('And just like that, you get notified about all your errors!')
```

CHAPTER

SIX

DOCUMENTATION

6.1 Changelog

6.1.1 (Unreleased)

6.1.2 1.2.0

- Added ability to cancel login to SMTP/GMAIL if credentials are used (#210, #266)
- Loosened dependencies (#209, #271)
- Added mimetype guessing for email (#239, #272)

6.1.3 1.0.4

- Added black and pre-commit
- Updated deps

6.1.4 1.0.0

- Added JSON Schema formatter support (#107)
- Improved documentation across the board

6.1.5 0.7.4

Maintenance release, broke markdown on pypi

6.1.6 0.7.3

Added

- Added ability to add email attachment via SMTP (#91) via (#99). Thanks @grabear
- Added direct notify ability via `notifiers.core.notify()` via (#101).

6.1.7 0.7.2

Added

- Mailgun support (#96)
- PopcornNotify support (#97)
- StatusPage.io support (#98)

Dependency changes

- Removed `requests-toolbelt` (it wasn't actually needed, `requests` was sufficient)

6.1.8 0.7.1

Maintenance release (added logo and donation link)

6.1.9 0.7.0

Added

- Pagerduty support (#95)
- Twilio support (#93)
- Added *Notification logger*

Note - For earlier changes please see Github releases

6.2 About

6.2.1 The problem

While I was working on a different project, I needed to enable its users to send notifications. Soon I discovered that this was a project by itself, discovering and implementing different provider API, testing it, relying on outdated documentation at times and etc. It was quite the endeavour. Some providers offered their own SDK packages, but that meant adding more dependencies to an already dependency rich project, which was not ideal. There has to be a better way, right?

6.2.2 The solution

Enter `notifiers`. A common interface to many, many notification providers, with a minimal set of dependencies.

6.2.3 The interface

Consistent naming

Right out of the gate there was an issue of consistent naming. Different API providers have different names for similar properties. For example, one provider can name its API key as `api_key`, another as `token`, the third as `apikey`

and etc. The solution I chose was to be as fateful to the original API properties as possible, with the only exception being the `message` property, which is shared among all notifiers and replaced internally as needed.

Snake Case

While the majority of providers already expect lower case and speicfically snake cased properties in their request, some do not. Notifiers normalizes this by making all request properties snake case and converting to relevant usage behind the scenes.

Reserved words issue

Some provider properties clash with python's reserved words, `from` being the most prominent example of that. There are two solutions to that issues. The first is to construct data via a dict and unpack it into the *`notify()`* command like so:

```
>>> data = {
...  'to': 'foo@bar.com',
...  'from': 'bar@foo.com'
... }
>>> provider.notify(**data)
```

The other is to use an alternate key word, which would always be the reservred key word followed by an underscore:

```
>>> provider.notify(to='foo@bar.com', from_='bar@foo.com')
```

6.2.4 What this is

A general wrapper for a variety of 3rd party providers and built in ones (like SMTP) aimed solely at sending notifications.

6.2.5 Who is this for

- Developers aiming to integrate 3rd party notifications into their application
- Script makes aiming to enable 3rd party notification abilities, either via python script or any CLI script
- Anyone that want to programmatically send notification and not be concerned about familiarizing themselves with 3rd party APIs or built in abilities

6.2.6 What this isn't

Many providers API enable many other capabilities other than sending notification. None of those will be handled by this module as it's outside its scope. If you need API access other than sending notification, look into implementing the 3rd party API directly, or using an SDK if available.

6.3 Installation

6.3.1 Via pip

You can install via pip:

```
$ pip install notifiers
```

Or install from source:

```
$ pip install https://github.com/notifiers/notifiers/master.zip
```

Use `develop` branch for cutting edge (not recommended):

```
$ pip install https://github.com/notifiers/notifiers/develop.zip
```

Note: Python 3.6 and higher is required when installing via pip

6.3.2 Via docker

Alternatively, use DockerHub:

```
$ docker pull liiight/notifiers
```

Use `develop` tag for cutting edge (still not recommended):

```
$ docker pull liiight/notifiers:develop
```

Or build from `DockerFile` locally

6.4 Usage

6.4.1 Basic Usage

The easiest way to initialize a notifier is via the *get_notifier()* helper:

```
>>> import notifiers
>>> pushover = notifiers.get_notifier('pushover')
```

Or import it directly:

```
>>> from notifiers.providers.pushover import Pushover
>>> pushover = Pushover()
```

To send a notification invoke the *notify()* method:

```
>>> pushover.notify(apikey='FOO', user='BAR', message='BAZ')
```

The *notifiers.core.Provider.notify()* method takes key word arguments based on the provider's schema. The `message` key word is used in all notifiers.

Note: You can also send a notification without getting a provider object via the *notifiers.core.notify()* method:

```
>>> from notifiers import notify
>>> notify('pushover', apikey='FOO', user='BAR', message='BAZ').
```

The first argument of the *notify()* method is the requested provider name. If such does not exist a `NoSuchNotifierError` exception will be raised.

If there's a problem with sent key words, a *NotifierException* will be thrown:

```
>>> import notifiers
>>> pushover = notifiers.get_notifier('pushover')
>>> pushover.notify(message='FOO')
Traceback (most recent call last):
  File "<input>", line 1, in <module>
  File "/Users/liiight/PycharmProjects/notifiers/notifiers/core.py", line 215, in␣
↪notify
    self._validate_data(kwargs, validator)
  File "/Users/liiight/PycharmProjects/notifiers/notifiers/core.py", line 193, in _
↪validate_data
    raise BadArguments(validation_error=msg, provider=self.name, data=data)
notifiers.exceptions.BadArguments: <NotificationError: Error with sent data: 'user'␣
↪is a required property>
```

In this case, a *BadArguments* exception was thrown since not all required key words were sent.

6.4.2 Provider schema

Notifier's schema is constructed with JSON Schema. Some understanding of it is needed in order to correctly construct the notification correctly. To see provider schema, use the `schema` property:

```
>>> pushover.schema
{'type': 'object', 'properties': {'user': {'oneOf': [{'type': 'array', 'items': {'type
↪': 'string', 'title': 'the user/group key (not e-mail address) of your user (or you)
↪'}, 'minItems': 1, 'uniqueItems': True}, {'type': 'string', 'title': 'the user/
↪group key (not e-mail address) of your user (or you)'}]}, 'message': {'type':
↪'string', 'title': 'your message'}, 'title': {'type': 'string', 'title': "your␣
↪message's title, otherwise your app's name is used"}, 'token': {'type': 'string',
↪'title': "your application's API token"}, 'device': {'oneOf': [{'type': 'array',
↪'items': {'type': 'string', 'title': "your user's device name to send the message␣
↪directly to that device"}, 'minItems': 1, 'uniqueItems': True}, {'type': 'string',
↪'title': "your user's device name to send the message directly to that device"}]},
↪'priority': {'type': 'number', 'minimum': -2, 'maximum': 2, 'title': 'notification␣
↪priority'}, 'url': {'type': 'string', 'format': 'uri', 'title': 'a supplementary␣
↪URL to show with your message'}, 'url_title': {'type': 'string', 'title': 'a title␣
↪for your supplementary URL, otherwise just the URL is shown'}, 'sound': {'type':
↪'string', 'title': "the name of one of the sounds supported by device clients to␣
↪override the user's default sound choice", 'enum': ['pushover', 'bike', 'bugle',
↪'cashregister', 'classical', 'cosmic', 'falling', 'gamelan', 'incoming',
↪'intermission', 'magic', 'mechanical', 'pianobar', 'siren', 'spacealarm', 'tugboat',
↪ 'alien', 'climb', 'persistent', 'echo', 'updown', 'none']}, 'timestamp': {'type':
↪'integer', 'minimum': 0, 'title': "a Unix timestamp of your message's date and time␣
↪to display to the user, rather than the time your message is received by our API"},
↪'retry': {'type': 'integer', 'minimum': 30, 'title': 'how often (in seconds) the␣
↪Pushover servers will send the same notification to the user. priority must be set␣
↪to 2'}, 'expire': {'type': 'integer', 'maximum': 86400, 'title': 'how many seconds␣
↪your notification will continue to be retried for. priority must be set to 2'},
↪'callback': {'type': 'string', 'format': 'uri', 'title': 'a publicly-accessible URL␣
↪that our servers will send a request to when the user has acknowledged your␣
↪notification. priority must be set to 2'}, 'html': {'type': 'integer', 'minimum': 0,
↪ 'maximum': 1, 'title': 'enable HTML formatting'}}, 'additionalProperties': False,
↪'required': ['user', 'message', 'token']}
```

(continues on next page)

(continued from previous page)

To see the required schema use the `required` property:

```
>>> pushover.required
{'required': ['user', 'message', 'token']}
```

The reply is always a dict which represent the validation of the schema. In this case it's pretty straightforward, but it can be more complex at times:

```
>>> hipchat = notifiers.get_notifier('hipchat')
>>> hipchat.required
{'allOf': [{'required': ['message', 'id', 'token']}, {'oneOf': [{'required': ['room']}
↪, {'required': ['user']}], 'error_oneOf': "Only one of 'room' or 'user' is allowed"}
↪, {'oneOf': [{'required': ['group']}, {'required': ['team_server']}], 'error_oneOf
↪': "Only one 'group' or 'team_server' is allowed"}]}
```

Hipchat's validation requires `message`, `id` and `token` are sent, exactly one of of `room` or `user` and exactly one of `group` or `team_server`.

To get all of the schema properties, which correlates to the key words it can handle, use `arguments`:

```
>>> pushover.arguments
{'user': {'oneOf': [{'type': 'array', 'items': {'type': 'string', 'title': 'the user/
↪group key (not e-mail address) of your user (or you)'}, 'minItems': 1, 'uniqueItems
↪': True}, {'type': 'string', 'title': 'the user/group key (not e-mail address) of␣
↪your user (or you)'}]}, 'message': {'type': 'string', 'title': 'your message'},
↪'title': {'type': 'string', 'title': "your message's title, otherwise your app's␣
↪name is used"}, 'token': {'type': 'string', 'title': "your application's API token"}
↪, 'device': {'oneOf': [{'type': 'array', 'items': {'type': 'string', 'title': "your␣
↪user's device name to send the message directly to that device"}, 'minItems': 1,
↪'uniqueItems': True}, {'type': 'string', 'title': "your user's device name to send␣
↪the message directly to that device"}]}, 'priority': {'type': 'number', 'minimum': -
↪2, 'maximum': 2, 'title': 'notification priority'}, 'url': {'type': 'string',
↪'format': 'uri', 'title': 'a supplementary URL to show with your message'}, 'url_
↪title': {'type': 'string', 'title': 'a title for your supplementary URL, otherwise␣
↪just the URL is shown'}, 'sound': {'type': 'string', 'title': "the name of one of␣
↪the sounds supported by device clients to override the user's default sound choice",
↪ 'enum': ['pushover', 'bike', 'bugle', 'cashregister', 'classical', 'cosmic',
↪'falling', 'gamelan', 'incoming', 'intermission', 'magic', 'mechanical', 'pianobar',
↪ 'siren', 'spacealarm', 'tugboat', 'alien', 'climb', 'persistent', 'echo', 'updown',
↪ 'none']}, 'timestamp': {'type': 'integer', 'minimum': 0, 'title': "a Unix␣
↪timestamp of your message's date and time to display to the user, rather than the␣
↪time your message is received by our API"}, 'retry': {'type': 'integer', 'minimum':␣
↪30, 'title': 'how often (in seconds) the Pushover servers will send the same␣
↪notification to the user. priority must be set to 2'}, 'expire': {'type': 'integer',
↪ 'maximum': 86400, 'title': 'how many seconds your notification will continue to be␣
↪retried for. priority must be set to 2'}, 'callback': {'type': 'string', 'format':
↪'uri', 'title': 'a publicly-accessible URL that our servers will send a request to␣
↪when the user has acknowledged your notification. priority must be set to 2'}, 'html
↪': {'type': 'integer', 'minimum': 0, 'maximum': 1, 'title': 'enable HTML formatting
↪'}}
```

6.4.3 Environment variables

You can set environment variable to replace any argument that the notifier can use. The default syntax to follow is `NOTIFIERS_[PROVIDER_NAME]_[ARGUMENT_NAME]`:

```
$ export NOTIFIERS_PUSHOVER_TOKEN=FOO
$ export NOTIFIERS_PUSHOVER_USER=BAR
```

Then you could just use:

```
>>> p.notify(message='message')
```

Note that you can also set `MESSAGE` in an environment variable. You can also change the default prefix of `NOTIFIERS_` by pass the `env_prefix` argument on notify:

```
>>> p.notify(message='test', env_prefix='MY_OWN_PREFIX_')
```

6.4.4 Provider resources

Some provider have helper method to enable fetching relevant resources (like rooms, users etc.) To get a list of provider resources use the *notifiers.core.Provider.resources()* property:

```
>>> telegram.resources
['updates']
```

Resource share almost all of their functionality with the *Provider* class, as they have a schema as well:

```
>>> telegram.updates
<ProviderResource,provider=telegram,resource=updates>
>>> telegram.updates.schema
{'type': 'object', 'properties': {'token': {'type': 'string', 'title': 'Bot token'}},
↪'additionalProperties': False, 'required': ['token']}
```

To invoke the resource, just call it:

```
>>> telegram.updates()
Traceback (most recent call last):
  File "<input>", line 1, in <module>
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 278, in __
↪call__
    data = self._process_data(**kwargs)
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 204, in _
↪process_data
    self._validate_data(data, validator)
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 175, in _
↪validate_data
    raise BadArguments(validation_error=msg, provider=self.name, data=data)
notifiers.exceptions.BadArguments: Error with sent data: 'token' is a required␣
↪property
```

Oops, forgot to send `token`. Let's try again:

```
>>> telegram.updates(token='foo')
[{'update_id': REDACTED, 'message': {'message_id': REDACTED, 'from': {'id': REDACTED,
↪'is_bot': False, 'first_name': 'REDACTED', 'last_name': 'REDACTED', 'username':
↪'REDACTED', 'language_code': 'en-US'}, 'chat': {'id': REDACTED, 'first_name':
↪'REDACTED', 'last_name': 'REDACTED', 'username': 'REDACTED', 'type': 'private'},
↪'date': 1516178366, 'text': 'Ccc'}}]
```

(continues on next page)

(continued from previous page)

As can be expected, each provider resource returns a completely different response that correlates to the underlying API command it wraps. In this example, by invoking the `notifiers.providers.telegram.Telegram.updates()` method, you get a response that shows you which active chat IDs your telegram bot token can send to.

6.4.5 Handling errors

There are two base types of errors in Notifiers, data (or schema) related errors and notification related errors. The former can present as so:

```
>>> import notifiers
>>> pushover = notifiers.get_notifier('pushover')
>>> pushover.notify(message='FOO')
Traceback (most recent call last):
  File "<input>", line 1, in <module>
  File "/Users/liiight/PycharmProjects/notifiers/notifiers/core.py", line 215, in␣
↪notify
    self._validate_data(kwargs, validator)
  File "/Users/liiight/PycharmProjects/notifiers/notifiers/core.py", line 193, in _
↪validate_data
    raise BadArguments(validation_error=msg, provider=self.name, data=data)
notifiers.exceptions.BadArguments: <NotificationError: Error with sent data: 'user'␣
↪is a required property>
```

Here we see that an `BadArguments` exception was raised instantly, since not all required values were sent. Another example:

```
>>> pushover.notify(message='FOO', token='TOKEN', user='USER', attachment='/foo')
Traceback (most recent call last):
  File "/Users/orcarmi/PycharmProjects/notifiers/poc.py", line 50, in <module>
    raise_on_errors=True)
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 273, in␣
↪notify
    data = self._process_data(**kwargs)
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 203, in _
↪process_data
    self._validate_data(data)
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 176, in _
↪validate_data
    raise BadArguments(validation_error=msg, provider=self.name, data=data)
notifiers.exceptions.BadArguments: Error with sent data: 'foo' is not a 'valid_file'
```

Some values have both `type` and `format` set in their schema, which enforces a specific logic. Here we can see that the schema for pushover's `attachment` attribute has `format` set to `valid_file` which check that the file is present.

There are also notification based errors:

```
>>> rsp = pushover.notify(message='FOO', token='BAD TOKEN', user='USER')
>>> rsp.raise_on_errors()
Traceback (most recent call last):
  File "/Users/orcarmi/PycharmProjects/notifiers/poc.py", line 49, in <module>
    raise_on_errors=True)
```

(continues on next page)

(continued from previous page)

```
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 276, in␣
↪notify
    rsp.raise_on_errors()
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 48, in␣
↪raise_on_errors
    raise NotificationError(provider=self.provider, data=self.data, errors=self.
↪errors, response=self.response)
notifiers.exceptions.NotificationError: Notification errors: application token is␣
↪invalid
```

Note the default behaviour for *Response* is not to raise exception on error. You can either use the *raise_on_errors()* method, or pass `raise_on_errors=True` to the notification command:

```
>>> pushover.notify(message='FOO', token='BAD TOKEN', user='USER', raise_on_
↪errors=True)
Traceback (most recent call last):
  File "/Users/orcarmi/PycharmProjects/notifiers/poc.py", line 49, in <module>
    raise_on_errors=True)
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 276, in␣
↪notify
    rsp.raise_on_errors()
  File "/Users/orcarmi/PycharmProjects/notifiers/notifiers/core.py", line 48, in␣
↪raise_on_errors
    raise NotificationError(provider=self.provider, data=self.data, errors=self.
↪errors, response=self.response)
notifiers.exceptions.NotificationError: Notification errors: application token is␣
↪invalid
```

You can also use the `ok` property:

```
>>> rsp = pushover.notify(message='FOO', token='BAD TOKEN', user='USER')
>>> rsp.ok
False
>>> rsp.errors
['application token is invalid']
```

6.5 Command Line Interface

Notifiers come with CLI support

6.5.1 Main view

To view the main help just enter `notifiers` or `notifiers --help`:

```
$ notifiers
Usage: notifiers [OPTIONS] COMMAND [ARGS]...

  Notifiers CLI operation

Options:
 --version          Show the version and exit.
 --env-prefix TEXT  Set a custom prefix for env vars usage
 --help             Show this message and exit.
```

(continues on next page)

(continued from previous page)

```
Commands:
  email       Options for 'email'
  gitter      Options for 'gitter'
  gmail       Options for 'gmail'
  hipchat     Options for 'hipchat'
  join        Options for 'join'
  providers   Shows all available providers
  pushbullet  Options for 'pushbullet'
  pushover    Options for 'pushover'
  simplepush  Options for 'simplepush'
  slack       Options for 'slack'
  telegram    Options for 'telegram'
  zulip       Options for 'zulip'
```

To view all providers use the `providers` command like so:

```
$ notifiers providers
pushover, simplepush, slack, email, gmail, telegram, gitter, pushbullet, join,␣
↪hipchat, zulip
```

This will return all available provider names

6.5.2 Provider actions

Each provider has a group of actions it can perform. Due to the generic nature that providers are implemented in, these actions are usually shared among all providers. To access available commands, use the `notifiers [PROVIDER_NAME] --help` command:

```
$ notifiers email --help
Usage: notifiers email [OPTIONS] COMMAND [ARGS]...

  Options for 'email'

Options:
  --help  Show this message and exit.

Commands:
  defaults  'email' default values
  metadata  'email' metadata
  notify    Send emails via SMTP
  required  'email' required schema
  schema    'email' full schema
```

The `defaults`, `metadata`, `required` and `schema` command all return a JSON output of the relevant provider property:

```
$ notifiers email metadata
{"base_url": null, "site_url": "https://en.wikipedia.org/wiki/Email", "provider_name
↪": "email"}
```

These helper methods can also accept a `--pretty` flag which will out a nicely indented JSON:

```
$ notifiers email metadata --pretty
{
    "base_url": null,
    "site_url": "https://en.wikipedia.org/wiki/Email",
    "provider_name": "email"
}
```

6.5.3 Sending a notification

To send a notification you use the `notify` command. Each notifier has its own set of relevant options it can take. View them by sending the `notifiers [PROVIDER_NAME] notify --help`:

```
$ notifiers email notify --help
Usage: notifiers email notify [OPTIONS] [MESSAGE]

  Send emails via SMTP

Options:
  --subject TEXT      The subject of the email message
  --to TEXT           One or more email addresses to use. Multiple usages of
                      this option are allowed
  --from TEXT         The from address to use in the email
  --host TEXT         The host of the smtp server
  --port INTEGER      The port number to use
  --username TEXT     Username if relevant
  --password TEXT     Password if relevant
  --tls / --no-tls    Should tls be used
  --ssl / --no-ssl    Should ssl be used
  --html / --no-html  Should the email be parse as an html file
  --help              Show this message and exit.
```

Note: Due to the nature of command line syntax, only primitive argument types can be used with it, meaning you can only pass string, int, float and booleans (using flags) when invoking the notify command via CLI. List and dict arguments cannot be passed with it.

Note that `message` is an expected argument that need to be either explicitly set or piped into the command.

6.5.4 Piping into a notification

Notifiers CLI enable using pipe to directly pass value to the `message` argument:

```
$ cat file.txt | notifiers notify email --to blah@foo.com
```

6.5.5 Environment variables

Environment variables are respected by all means of notification by notifiers and the CLI is no different to that aspect. If you defined for example `NOTIFIERS_PUSHOVER_TOKEN` and `NOTIFIERS_PUSHOVER_USER` you can simply run:

```
$ export NOTIFIERS_PUSHOVER_TOKEN=FOO
$ export NOTIFIERS_PUSHOVER_USER=BAR
$ notifiers notify pushover "wow, this is easy!"
```

You can change the default env var prefix (which is `NOTIFIERS_`) by sending the `--env-prefix` option:

```
$ notifiers --env-prefix FOO_ notify pushover "Yep, easy stuff!"
```

Note: You can create a convenience alias for your used provider to even simplify this further:

```
$ alias notify="notifiers notify pushover"
```

And when combining this with setting environment variables, you can run:

```
$ notify "this is even easier!"
```

6.5.6 Provider resources

Some providers have resource helper commands:

```
$ notifiers telegram resources
updates
```

You can also see them in the provider `--help` view:

```
$ notifiers telegram --help
Usage: notifiers telegram [OPTIONS] COMMAND [ARGS]...

  Options for 'telegram'

Options:
  --help  Show this message and exit.

Commands:
  defaults   'telegram' default values
  metadata   'telegram' metadata
  notify     Send Telegram notifications
  required   'telegram' required schema
  resources  Show provider resources list
  schema     'telegram' full schema
  updates    Return Telegram bot updates, correlating to...
```

These resources have their own option they can use:

```
$ notifiers telegram updates --help
Usage: notifiers telegram updates [OPTIONS]

  Return Telegram bot updates, correlating to the `getUpdates` method.
  Returns chat IDs needed to notifications

Options:
  --token TEXT             Bot token
  --pretty / --not-pretty  Output a pretty version of the JSON
  --help                   Show this message and exit.
```

Invoking them returns a JSON reply (usually), where each reply correlates to the API data.

Note: Like always, these resources play very nicely with environment variables, so if you set your token in an environment variable, the resource can pick that up by default

6.5.7 Version

Get installed `notifiers` version via the `--version` flag:

```
$ notifiers --version
notifiers 0.6.3
```

6.6 Notification logger

Notifiers enable you to log directly to a notifier via a stdlib logging handler, *`NotificationHandler`*:

```
>>> import logging
>>> from notifiers.logging import NotificationHandler

>>> log = logging.getLogger(__name__)
>>> defaults = {
...     'token': 'foo,
...     'user': 'bar
... }

>>> hdlr = NotificationHandler('pushover', defaults=defaults)
>>> hdlr.setLevel(logging.ERROR)

>>> log.addHandler(hdlr)
>>> log.error('And just like that, you get notified about all your errors!')
```

By setting the handler level to the desired one, you can directly get notified about relevant event in your application, without needing to change a single line of code.

6.6.1 Using environs

Like any other usage of notifiers, you can pass any relevant provider arguments via *Environment variables*.

6.6.2 Fallback notifiers

If you rely on 3rd party notifiers to send you notification about errors, you may want to have a fallback in case those notification fail for any reason. You can define a fallback notifier like so:

```
>>> fallback_defaults = {
...     'host': 'http://localhost,
...     'port': 80,
...     'username': 'foo',
...     'password': 'bar
... }
```

(continues on next page)

(continued from previous page)

```
>>> hdlr = NotificationHandler('pushover', fallback='email', fallback_
↪defaults=fallback_defaults)
```

Then in case there is an error with the main notifier, `pushover` in this case, you'll get a notification sent via `email`.

Note: *`NotificationHandler`* respect the standard `logging` `raiseExceptions` flag to determine if fallback should be used. Also, fallback is used only when any subclass of *`NotifierException`* occurs.

CHAPTER

SEVEN

PROVIDERS DOCUMENTATION

7.1 Providers

7.1.1 Email (SMTP)

Enables sending email messages to SMTP servers.

```
>>> from notifiers import get_notifier
>>> email = get_notifier('email')
>>> email.required
{'required': ['message', 'to']}

>>> email.notify(to='email@addrees.foo', message='hi!')
```

It uses several defaults:

```
>>> email.defaults
{'subject': "New email from 'notifiers'!", 'from': '[USER@HOSTNAME]', 'host':
↪'localhost', 'port': 25, 'tls': False, 'ssl': False, 'html': False}
```

Any of these can be overridden by sending them to the `notify()` command.

Full schema:

```
additionalProperties: false
dependencies:
  password:
  - username
  ssl:
  - tls
  username:
  - password
properties:
  attachments:
    oneOf:
    - items:
        format: valid_file
        title: one or more attachments to use in the email
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - format: valid_file
```

(continues on next page)

(continued from previous page)

```
      title: one or more attachments to use in the email
      type: string
  from:
    format: email
    title: the FROM address to use in the email
    type: string
  from_:
    duplicate: true
    format: email
    title: the FROM address to use in the email
    type: string
  host:
    format: hostname
    title: the host of the SMTP server
    type: string
  html:
    title: should the email be parse as an HTML file
    type: boolean
  message:
    title: the content of the email message
    type: string
  password:
    title: password if relevant
    type: string
  port:
    format: port
    title: the port number to use
    type: integer
  ssl:
    title: should SSL be used
    type: boolean
  subject:
    title: the subject of the email message
    type: string
  tls:
    title: should TLS be used
    type: boolean
  to:
    oneOf:
    - items:
        format: email
        title: one or more email addresses to use
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - format: email
      title: one or more email addresses to use
      type: string
  username:
    title: username if relevant
    type: string
  login:
    title: Trigger login to server
    type: boolean
required:
- message
```

(continues on next page)

(continued from previous page)

```
- to
```

type: object

7.1.2 Gitter

Send notifications via Gitter

```
>>> from notifiers import get_notifier
>>> gitter = get_notifier('gitter')
>>> gitter.required
{'required': ['message', 'token', 'room_id']}

>>> gitter.notify(message='Hi!', token='SECRET_TOKEN', room_id=1234)
```

Full schema:

```
additionalProperties: false
properties:
  message:
    title: Body of the message
    type: string
  room_id:
    title: ID of the room to send the notification to
    type: string
  token:
    title: access token
    type: string
required:
- message
- token
- room_id
```

type: object

You can view the available rooms you can access via the `rooms` resource

```
>>> gitter.rooms(token="SECRET_TOKEN")
{'id': '...', 'name': 'Foo/bar', ... }
```

7.1.3 Gmail

Send emails via Gmail

This is a private use case of the *SMTP* provider

```
>>> from notifiers import get_notifier
>>> gmail = get_notifier('gmail')
>>> gmail.defaults
{'subject': "New email from 'notifiers'!", 'from': '<USERNAME@HOST>', 'host': 'smtp.
↪gmail.com', 'port': 587, 'tls': True, 'ssl': False, 'html': False}

>>> gmail.notify(to='email@addrees.foo', message='hi!')
```

7.1.4 Hipchat

Send notification to Hipchat rooms

Simple example:

```
>>> from notifiers import get_notifier
>>> hipchat = get_notifier('hipchat')
>>> hipchat.notify(token='SECRET', group='foo', message='hi!', room=1234)
```

Hipchat requires using either a `group` or a `team_server` key word (for private instances)

You can view the users you can send to via the `users` resource:

```
>>> hipchat.users(token='SECRET', group='foo')
{'items': [{'id': 1, 'links': {'self': '...'}, 'mention_name': '...', 'name': '...',
↪'version': 'E4GX9340'}, ...]}
```

You can view the rooms you can send to via the `rooms` resource:

```
>>> hipchat.rooms(token='SECRET', group='foo')
{'items': [{'id': 9, 'is_archived': False, ... }]
```

Full schema:

```
additionalProperties: false
allOf:
- required:
  - message
  - id
  - token
- error_oneOf: Only one of 'room' or 'user' is allowed
  oneOf:
  - required:
    - room
  - required:
    - user
- error_oneOf: Only one 'group' or 'team_server' is allowed
  oneOf:
  - required:
    - group
  - required:
    - team_server
properties:
  attach_to:
    title: The message id to to attach this notification to
    type: string
  card:
    additionalProperties: false
    properties:
      activity:
        additionalProperties: false
        properties:
          html:
            title: Html for the activity to show in one line a summary of the action
              that happened
            type: string
          icon:
```

(continues on next page)

(continued from previous page)

```
          oneOf:
          - title: The url where the icon is
            type: string
          - additionalProperties: false
            properties:
              url:
                title: The url where the icon is
                type: string
              url@2x:
                title: The url for the icon in retina
                type: string
            required:
            - url
            type: object
      required:
      - html
      type: object
    attributes:
      items:
        additionalProperties: false
        properties:
          label:
            maxLength: 50
            minLength: 1
            title: Attribute label
            type: string
          value:
            properties:
              icon:
                oneOf:
                - title: The url where the icon is
                  type: string
                - additionalProperties: false
                  properties:
                    url:
                      title: The url where the icon is
                      type: string
                    url@2x:
                      title: The url for the icon in retina
                      type: string
                  required:
                  - url
                  type: object
              label:
                title: The text representation of the value
                type: string
              style:
                enum:
                - lozenge-success
                - lozenge-error
                - lozenge-current
                - lozenge-complete
                - lozenge-moved
                - lozenge
                title: AUI Integrations for now supporting only lozenges
                type: string
              url:
```

(continues on next page)

(continued from previous page)

```
                  title: Url to be opened when a user clicks on the label
                  type: string
              type: object
          required:
          - label
          - value
          type: object
        title: List of attributes to show below the card
        type: array
      description:
        oneOf:
        - type: string
        - additionalProperties: false
          properties:
            format:
              enum:
              - text
              - html
              title: Determines how the message is treated by our server and rendered
                inside HipChat applications
              type: string
            value:
              maxLength: 1000
              minLength: 1
              type: string
          required:
          - value
          - format
          type: object
      format:
        enum:
        - compact
        - medium
        title: Application cards can be compact (1 to 2 lines) or medium (1 to 5␣
↪lines)
        type: string
      style:
        enum:
        - file
        - image
        - application
        - link
        - media
        title: Type of the card
        type: string
      thumbnail:
        additionalProperties: false
        properties:
          height:
            title: The original height of the image
            type: integer
          url:
            maxLength: 250
            minLength: 1
            title: The thumbnail url
            type: string
          url@2x:
```

(continues on next page)

(continued from previous page)

```
            maxLength: 250
            minLength: 1
            title: The thumbnail url in retina
            type: string
          width:
            title: The original width of the image
            type: integer
        required:
        - url
        type: object
      title:
        maxLength: 500
        minLength: 1
        title: The title of the card
        type: string
      url:
        title: The url where the card will open
        type: string
    required:
    - style
    - title
    type: object
  color:
    enum:
    - yellow
    - green
    - red
    - purple
    - gray
    - random
    title: Background color for message
    type: string
  from:
    title: A label to be shown in addition to the sender's name
    type: string
  group:
    title: HipChat group name
    type: string
  icon:
    oneOf:
    - title: The url where the icon is
      type: string
    - additionalProperties: false
      properties:
        url:
          title: The url where the icon is
          type: string
        url@2x:
          title: The url for the icon in retina
          type: string
      required:
      - url
      type: object
  id:
    title: An id that will help HipChat recognise the same card when it is sent␣
↪multiple
      times
```

(continues on next page)

(continued from previous page)

```
    type: string
  message:
    maxLength: 10000
    minLength: 1
    title: The message body
    type: string
  message_format:
    enum:
    - text
    - html
    title: Determines how the message is treated by our server and rendered inside
      HipChat applications
    type: string
  notify:
    title: Whether this message should trigger a user notification (change the tab
      color, play a sound, notify mobile phones, etc). Each recipient's notification
      preferences are taken into account.
    type: boolean
  room:
    maxLength: 100
    minLength: 1
    title: The id or url encoded name of the room
    type: string
  team_server:
    title: 'An alternate team server. Example: ''https://hipchat.corp-domain.com'''
    type: string
  token:
    title: User token
    type: string
  user:
    title: The id, email address, or mention name (beginning with an '@') of the user
      to send a message to.
    type: string
type: object
```

7.1.5 Join

Send notification via Join

```
>>> from notifiers import get_notifier
>>> join = get_notifier('join')
>>> join.notify(apikey='SECRET', message='Hi!')
```

You can view the devices you can send to via the `devices` resource:

```
>>> join.devices(apikey='SECRET')
{'items': [{'id': 9, 'is_archived': False, ... }]
```

Full schema:

```
additionalProperties: false
anyOf:
- dependencies:
    smsnumber:
    - smstext
```

(continues on next page)

(continued from previous page)

```
- dependencies:
    smsnumber:
    - mmsfile
dependencies:
  callnumber:
  - smsnumber
  smstext:
  - smsnumber
error_anyOf: Must use either 'smstext' or 'mmsfile' with 'smsnumber'
properties:
  alarmVolume:
    title: set device alarm volume
    type: string
  apikey:
    title: user API key
    type: string
  callnumber:
    title: number to call to
    type: string
  clipboard:
    title: "some text you want to set on the receiving device\u2019s clipboard"
    type: string
  deviceId:
    title: The device ID or group ID of the device you want to send the message to
    type: string
  deviceIds:
    oneOf:
    - items:
        title: A comma separated list of device IDs you want to send the push to
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - title: A comma separated list of device IDs you want to send the push to
      type: string
  deviceNames:
    oneOf:
    - items:
        title: A comma separated list of device names you want to send the push to
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - title: A comma separated list of device names you want to send the push to
      type: string
  file:
    format: uri
    title: a publicly accessible URL of a file
    type: string
  find:
    title: set to true to make your device ring loudly
    type: boolean
  group:
    title: allows you to join notifications in different groups
    type: string
  icon:
    format: uri
```

(continues on next page)

(continued from previous page)

```
    title: notification's icon URL
    type: string
  image:
    format: uri
    title: Notification image URL
    type: string
  interruptionFilter:
    maximum: 4
    minimum: 1
    title: set interruption filter mode
    type: integer
  mediaVolume:
    title: set device media volume
    type: integer
  message:
    title: usually used as a Tasker or EventGhost command. Can also be used with URLs
      and Files to add a description for those elements
    type: string
  mmsfile:
    format: uri
    title: publicly accessible mms file url
    type: string
  priority:
    maximum: 2
    minimum: -2
    title: control how your notification is displayed
    type: integer
  ringVolume:
    title: set device ring volume
    type: string
  smallicon:
    format: uri
    title: Status Bar Icon URL
    type: string
  smsnumber:
    title: phone number to send an SMS to
    type: string
  smstext:
    title: some text to send in an SMS
    type: string
  title:
    title: "If used, will always create a notification on the receiving device with\
      \ this as the title and text as the notification\u2019s text"
    type: string
  url:
    format: uri
    title: ' A URL you want to open on the device. If a notification is created with
      this push, this will make clicking the notification open this URL'
    type: string
  wallpaper:
    format: uri
    title: a publicly accessible URL of an image file
    type: string
required:
- apikey
- message
type: object
```

7.1.6 Mailgun

Send notification via Mailgun

```
>>> from notifiers import get_notifier
>>> mailgun = get_notifiers('mailgun')
>>> mailgun.notify(to='foo@bar.baz', domain='mydomain', api_key='SECRET', message='Hi!
↪')
```

Full schema:

```
additionalProperties: false
allOf:
- required:
  - to
  - domain
  - api_key
- anyOf:
  - required:
    - from
  - required:
    - from_
- anyOf:
  - required:
    - message
  - required:
    - html
  error_anyOf: Need either "message" or "html"
properties:
  api_key:
    title: User's API key
    type: string
  attachment:
    oneOf:
    - items:
        format: valid_file
        title: File attachment
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - format: valid_file
      title: File attachment
      type: string
  bcc:
    oneOf:
    - items:
        title: 'Email address of the recipient(s). Example: "Bob <bob@host.com>".'
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - title: 'Email address of the recipient(s). Example: "Bob <bob@host.com>".'
      type: string
  cc:
    oneOf:
    - items:
        title: 'Email address of the recipient(s). Example: "Bob <bob@host.com>".'
```

(continues on next page)

(continued from previous page)

```
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - title: 'Email address of the recipient(s). Example: "Bob <bob@host.com>".'
      type: string
  data:
    additionalProperties:
      type: object
    title: attach a custom JSON data to the message
    type: object
  deliverytime:
    format: rfc2822
    title: 'Desired time of delivery. Note: Messages can be scheduled for a maximum
      of 3 days in the future.'
    type: string
  dkim:
    title: Enables/disables DKIM signatures on per-message basis
    type: boolean
  domain:
    title: MailGun's domain to use
    type: string
  from:
    format: email
    title: Email address for From header
    type: string
  from_:
    duplicate: true
    format: email
    title: Email address for From header
    type: string
  headers:
    additionalProperties:
      type: string
    title: Any other header to add
    type: object
  html:
    title: Body of the message. (HTML version)
    type: string
  inline:
    oneOf:
    - items:
        format: valid_file
        title: Attachment with inline disposition. Can be used to send inline images
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - format: valid_file
      title: Attachment with inline disposition. Can be used to send inline images
      type: string
  message:
    title: Body of the message. (text version)
    type: string
  require_tls:
    title: If set to True this requires the message only be sent over a TLS␣
↪connection.
```

(continues on next page)

(continued from previous page)

```
      If a TLS connection can not be established, Mailgun will not deliver the␣
↪message.If
      set to False, Mailgun will still try and upgrade the connection, but if Mailgun
      can not, the message will be delivered over a plaintext SMTP connection.
    type: boolean
  skip_verification:
    title: If set to True, the certificate and hostname will not be verified when
      trying to establish a TLS connection and Mailgun will accept any certificate
      during delivery. If set to False, Mailgun will verify the certificate and␣
↪hostname.
      If either one can not be verified, a TLS connection will not be established.
    type: boolean
  subject:
    title: Message subject
    type: string
  tag:
    oneOf:
    - items:
        format: ascii
        maxLength: 128
        title: Tag string
        type: string
      maxItems: 3
      minItems: 1
      type: array
      uniqueItems: true
    - format: ascii
      maxLength: 128
      title: Tag string
      type: string
  testmode:
    title: Enables sending in test mode.
    type: boolean
  to:
    oneOf:
    - items:
        title: 'Email address of the recipient(s). Example: "Bob <bob@host.com>".'
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - title: 'Email address of the recipient(s). Example: "Bob <bob@host.com>".'
      type: string
  tracking:
    title: Toggles tracking on a per-message basis
    type: boolean
  tracking_clicks:
    enum:
    - true
    - false
    - htmlonly
    title: Toggles clicks tracking on a per-message basis. Has higher priority than
      domain-level setting. Pass yes, no or htmlonly.
    type:
    - string
    - boolean
  tracking_opens:
```

(continues on next page)

(continued from previous page)

```
    title: Toggles opens tracking on a per-message basis. Has higher priority than
      domain-level setting
    type: boolean
type: object
```

7.1.7 Pagerduty

Open Pagerduty incidents

```
>>> from notifiers import get_notifier
>>> pagerduty = get_notifier('pagerduty')
>>> pagerduty.notify(
...     message='Oh oh...',
...     event_action='trigger',
...     source='prod',
...     severity='info'
... )
```

Full schema:

```
properties:
  class:
    title: The class/type of the event
    type: string
  component:
    title: Component of the source machine that is responsible for the event
    type: string
  custom_details:
    title: Additional details about the event and affected system
    type: object
  dedup_key:
    maxLength: 255
    title: Deduplication key for correlating triggers and resolves
    type: string
  event_action:
    enum:
    - trigger
    - acknowledge
    - resolve
    title: The type of event
    type: string
  group:
    title: Logical grouping of components of a service
    type: string
  images:
    items:
      additionalProperties: false
      properties:
        alt:
          title: Optional alternative text for the image
          type: string
        href:
          title: Optional URL; makes the image a clickable link
          type: string
        src:
```

(continues on next page)

(continued from previous page)

```
          title: The source of the image being attached to the incident. This image
            must be served via HTTPS.
          type: string
      required:
      - src
      type: object
    type: array
  links:
    items:
      additionalProperties: false
      properties:
        href:
          title: URL of the link to be attached
          type: string
        text:
          title: Plain text that describes the purpose of the link, and can be used
            as the link's text
          type: string
      required:
      - href
      - text
      type: object
    type: array
  message:
    title: A brief text summary of the event, used to generate the summaries/titles
      of any associated alerts
    type: string
  routing_key:
    title: The GUID of one of your Events API V2 integrations. This is the
↪"Integration
      Key" listed on the Events API V2 integration's detail page
    type: string
  severity:
    enum:
    - critical
    - error
    - warning
    - info
    title: The perceived severity of the status the event is describing with respect
      to the affected system
    type: string
  source:
    title: The unique location of the affected system, preferably a hostname or FQDN
    type: string
  timestamp:
    format: iso8601
    title: The time at which the emitting tool detected or generated the event in
      ISO 8601
    type: string
required:
- routing_key
- event_action
- source
- severity
- message
type: object
```

7.1.8 PopcornNotify

Send PopcornNotify notifications

```
>>> from notifiers import get_notifier
>>> popcornnotify = get_notifier('popcornnotify')
>>> popcornnotify.notify(
...     message='Hi!',
...     api_key='SECRET',
...     recipients=[
...         'foo@bar.com',
...     ],
...     subject='Message subject!'
... )
```

Full schema:

```
properties:
  api_key:
    title: The API key
    type: string
  message:
    title: The message to send
    type: string
  recipients:
    oneOf:
    - items:
        format: email
        title: The recipient email address or phone number. Or an array of email␣
↪addresses
          and phone numbers
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - format: email
      title: The recipient email address or phone number. Or an array of email␣
↪addresses
        and phone numbers
      type: string
  subject:
    title: The subject of the email. It will not be included in text messages.
    type: string
required:
- message
- api_key
- recipients
type: object
```

7.1.9 Pushbullet

Send Pushbullet notifications.

```
>>> from notifiers import get_notifier
>>> pushbullet = get_notifier('pushbullet')
>>> pushbullet.notify(
```

(continues on next page)

(continued from previous page)

```
...     message='Hi!',
...     token='SECRET',
...     title='Message title',
...     type_='note',
...     url='https://url.in/message',
...     source_device_iden='FOO',
...     device_iden='bar',
...     client_iden='baz',
...     channel_tag='channel tag',
...     email='foo@bar.com',
...     guid='1234abcd',
... )
```

You can view the devices you can send to via the `devices` resource:

```
>>> pushbullet.devices(token='SECRET')
[{'active': True, 'iden': ... }]
```

Full schema:

```
additionalProperties: false
properties:
  channel_tag:
    title: Channel tag of the target channel, sends a push to all people who are␣
↪subscribed
      to this channel. The current user must own this channel.
    type: string
  client_iden:
    title: Client iden of the target client, sends a push to all users who have␣
↪granted
      access to this client. The current user must own this client
    type: string
  device_iden:
    title: Device iden of the target device, if sending to a single device
    type: string
  email:
    format: email
    title: Email address to send the push to. If there is a pushbullet user with this
      address, they get a push, otherwise they get an email
    type: string
  guid:
    title: Unique identifier set by the client, used to identify a push in case you
      receive it from /v2/everything before the call to /v2/pushes has completed.
      This should be a unique value. Pushes with guid set are mostly idempotent,␣
↪meaning
      that sending another push with the same guid is unlikely to create another push
      (it will return the previously created push).
    type: string
  message:
    title: Body of the push
    type: string
  source_device_iden:
    title: Device iden of the sending device
    type: string
  title:
    title: Title of the push
    type: string
```

(continues on next page)

(continued from previous page)

```
  token:
    title: API access token
    type: string
  type:
    enum:
    - note
    - link
    title: Type of the push, one of "note" or "link"
    type: string
  type_:
    enum:
    - note
    - link
    title: Type of the push, one of "note" or "link"
    type: string
  url:
    title: URL field, used for type="link" pushes
    type: string
required:
- message
- token
type: object
```

7.1.10 Pushover

Send Pushover notifications

Minimal example:

```
>>> from notifiers import get_notifier
>>> pushover = get_notifier('pushover')
>>> pushover.notify(message='Hi!', user='USER', token='TOKEN')
```

You can view the sounds you can user in the notification via the `sounds` resource:

```
>>> pushover.sounds(token='SECRET')
['pushover', 'bike', 'bugle', 'cashregister', 'classical', 'cosmic', 'falling',
↪'gamelan', 'incoming', 'intermission', 'magic', 'mechanical', 'pianobar', 'siren',
↪'spacealarm', 'tugboat', 'alien', 'climb', 'persistent', 'echo', 'updown', 'none']
```

You can view the limits of your application token using the `limits` resource:

```
>>> pushover.limits(token='SECRET')
{'limit': 7500, 'remaining': 6841, 'reset': 1535778000, 'status': 1, 'request':
↪'f0cb73b1-810d-4b9a-b275-394481bceb74'}
```

Full schema:

```
additionalProperties: false
properties:
  attachment:
    format: valid_file
    title: an image attachment to send with the message
    type: string
  callback:
```

(continues on next page)

(continued from previous page)

```
    format: uri
    title: a publicly-accessible URL that our servers will send a request to when
      the user has acknowledged your notification. priority must be set to 2
    type: string
  device:
    oneOf:
    - items:
        title: your user's device name to send the message directly to that device
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - title: your user's device name to send the message directly to that device
      type: string
  expire:
    maximum: 86400
    title: how many seconds your notification will continue to be retried for.␣
↪priority
      must be set to 2
    type: integer
  html:
    title: enable HTML formatting
    type: boolean
  message:
    title: your message
    type: string
  priority:
    maximum: 2
    minimum: -2
    title: notification priority
    type: integer
  retry:
    minimum: 30
    title: how often (in seconds) the Pushover servers will send the same notification
      to the user. priority must be set to 2
    type: integer
  sound:
    title: the name of one of the sounds supported by device clients to override the
      user's default sound choice. See `sounds` resource
    type: string
  timestamp:
    format: timestamp
    minimum: 0
    title: a Unix timestamp of your message's date and time to display to the user,
      rather than the time your message is received by our API
    type:
    - integer
    - string
  title:
    title: your message's title, otherwise your app's name is used
    type: string
  token:
    title: your application's API token
    type: string
  url:
    format: uri
    title: a supplementary URL to show with your message
```

(continues on next page)

(continued from previous page)

```
    type: string
  url_title:
    title: a title for your supplementary URL, otherwise just the URL is shown
    type: string
  user:
    oneOf:
    - items:
        title: the user/group key (not e-mail address) of your user (or you)
        type: string
      minItems: 1
      type: array
      uniqueItems: true
    - title: the user/group key (not e-mail address) of your user (or you)
      type: string
required:
- user
- message
- token
type: object
```

7.1.11 SimplePush

Send SimplePush notifications

Minimal example:

```
>>> from notifiers import get_notifier
>>> simplepush = get_notifier('simplepush')
>>> simplepush.notify(message='Hi!', key='KEY')
```

Full schema:

```
additionalProperties: false
properties:
  event:
    title: Event ID
    type: string
  key:
    title: your user key
    type: string
  message:
    title: your message
    type: string
  title:
    title: message title
    type: string
required:
- key
- message
type: object
```

7.1.12 Slack (Webhooks)

Send Slack webhook notifications.

Minimal example:

```
>>> from notifiers import get_notifier
>>> slack = get_notifier('slack')
>>> slack.notify(message='Hi!', webhook_url='https://url.to/webhook')
```

Full schema:

```
additionalProperties: false
properties:
  attachments:
    items:
      additionalProperties: false
      properties:
        author_icon:
          title: A valid URL that displays a small 16x16px image to the left of the
            author_name text. Will only work if author_name is present
          type: string
        author_link:
          title: A valid URL that will hyperlink the author_name text mentioned above.
            Will only work if author_name is present
          type: string
        author_name:
          title: Small text used to display the author's name
          type: string
        color:
          title: Can either be one of 'good', 'warning', 'danger', or any hex color
            code
          type: string
        fallback:
          title: A plain-text summary of the attachment. This text will be used in
            clients that don't show formatted text (eg. IRC, mobile notifications)
            and should not contain any markup.
          type: string
        fields:
          items:
            additionalProperties: false
            properties:
              short:
                title: Optional flag indicating whether the `value` is short enough
                  to be displayed side-by-side with other values
                type: boolean
              title:
                title: Required Field Title
                type: string
              value:
                title: Text value of the field. May contain standard message markup
                  and must be escaped as normal. May be multi-line
                type: string
            required:
            - title
            type: object
          minItems: 1
          title: Fields are displayed in a table on the message
          type: array
        footer:
          title: Footer text
          type: string
```

(continues on next page)

(continued from previous page)

```
        footer_icon:
          format: uri
          title: Footer icon URL
          type: string
        image_url:
          format: uri
          title: Image URL
          type: string
        pretext:
          title: Optional text that should appear above the formatted data
          type: string
        text:
          title: Optional text that should appear within the attachment
          type: string
        thumb_url:
          format: uri
          title: Thumbnail URL
          type: string
        title:
          title: Attachment title
          type: string
        title_link:
          title: Attachment title URL
          type: string
        ts:
          format: timestamp
          title: Provided timestamp (epoch)
          type:
          - integer
          - string
      required:
      - fallback
      type: object
    type: array
  channel:
    title: override default channel or private message
    type: string
  icon_emoji:
    title: override bot icon with emoji name.
    type: string
  icon_url:
    format: uri
    title: override bot icon with image URL
    type: string
  message:
    title: This is the text that will be posted to the channel
    type: string
  unfurl_links:
    title: avoid automatic attachment creation from URLs
    type: boolean
  username:
    title: override the displayed bot name
    type: string
  webhook_url:
    format: uri
    title: the webhook URL to use. Register one at https://my.slack.com/services/new/
↪incoming-webhook/
```

(continues on next page)

(continued from previous page)

```
    type: string
required:
- webhook_url
- message
type: object
```

7.1.13 StatusPage

Send StatusPage.io notifications

Minimal example:

```
>>> from notifiers import get_notifier
>>> statuspage = get_notifier('statuspage')
>>> statuspage.notify(message='Hi!', api_key='KEY', page_id='123ABC')
```

You can view the components you use in the notification via the `components` resource:

```
>>> statuspage.components(api_key='KEY', page_id='123ABC')
[{'id': '...', 'page_id': '...', ...]
```

Full schema:

```
additionalProperties: false
dependencies:
  backfill_date:
  - backfilled
  backfilled:
  - backfill_date
  scheduled_auto_completed:
  - scheduled_for
  scheduled_auto_in_progress:
  - scheduled_for
  scheduled_for:
  - scheduled_until
  scheduled_remind_prior:
  - scheduled_for
  scheduled_until:
  - scheduled_for
properties:
  api_key:
    title: OAuth2 token
    type: string
  backfill_date:
    format: date
    title: Date of incident in YYYY-MM-DD format
    type: string
  backfilled:
    title: Create an historical incident
    type: boolean
  body:
    title: The initial message, created as the first incident update
    type: string
  component_ids:
    items:
```

(continues on next page)

(continued from previous page)

```
      type: string
    title: List of components whose subscribers should be notified (only applicable
      for pages with component subscriptions enabled)
    type: array
  deliver_notifications:
    title: Control whether notifications should be delivered for the initial incident
      update
    type: boolean
  impact_override:
    enum:
    - none
    - minor
    - major
    - critical
    title: Override calculated impact value
    type: string
  message:
    title: The name of the incident
    type: string
  page_id:
    title: Page ID
    type: string
  scheduled_auto_completed:
    title: Automatically transition incident to 'Completed' at end
    type: boolean
  scheduled_auto_in_progress:
    title: Automatically transition incident to 'In Progress' at start
    type: boolean
  scheduled_for:
    format: iso8601
    title: Time the scheduled maintenance should begin
    type: string
  scheduled_remind_prior:
    title: Remind subscribers 60 minutes before scheduled start
    type: boolean
  scheduled_until:
    format: iso8601
    title: Time the scheduled maintenance should end
    type: string
  status:
    enum:
    - investigating
    - identified
    - monitoring
    - resolved
    - scheduled
    - in_progress
    - verifying
    - completed
    title: Status of the incident
    type: string
  wants_twitter_update:
    title: Post the new incident to twitter
    type: boolean
required:
- message
- api_key
```

(continues on next page)

(continued from previous page)

```
- page_id
type: object
```

7.1.14 Telegram

Send Telegram notifications.

Minimal example:

```
>>> from notifiers import get_notifier
>>> telegram = get_notifier('telegram')
>>> telegram.notify(message='Hi!', token='TOKEN', chat_id=1234)
```

You can view the available updates you can access via the `updates` resource

```
>>> telegram.updates(token="SECRET_TOKEN")
{'id': '...', 'name': 'Foo/bar', ... }
```

Full schema:

```
additionalProperties: false
properties:
  chat_id:
    oneOf:
    - type: string
    - type: integer
    title: Unique identifier for the target chat or username of the target channel
      (in the format @channelusername)
  disable_notification:
    title: Sends the message silently. Users will receive a notification with no␣
↪sound.
    type: boolean
  disable_web_page_preview:
    title: Disables link previews for links in this message
    type: boolean
  message:
    title: Text of the message to be sent
    type: string
  parse_mode:
    enum:
    - markdown
    - html
    title: Send Markdown or HTML, if you want Telegram apps to show bold, italic,
      fixed-width text or inline URLs in your bot's message.
    type: string
  reply_to_message_id:
    title: If the message is a reply, ID of the original message
    type: integer
  token:
    title: Bot token
    type: string
required:
- message
- chat_id
- token
type: object
```

7.1.15 Twilio

Send Twilio SMS

```
>>> from notifiers import get_notifier
>>> twilio = get_notifier('twilio')
>>> twilio.notify(message='Hi!', to='+12345678', account_sid=1234, auth_token='TOKEN')
```

Full schema:

```
allOf:
- anyOf:
  - anyOf:
    - required:
      - from
    - required:
      - from_
  - required:
    - messaging_service_id
  error_anyOf: Either 'from' or 'messaging_service_id' are required
- anyOf:
  - required:
    - message
  - required:
    - media_url
  error_anyOf: Either 'message' or 'media_url' are required
- required:
  - to
  - account_sid
  - auth_token
properties:
  account_sid:
    title: The unique id of the Account that sent this message.
    type: string
  application_sid:
    title: Twilio will POST MessageSid as well as MessageStatus=sent or␣
↪MessageStatus=failed
      to the URL in the MessageStatusCallback property of this Application
    type: string
  auth_token:
    title: The user's auth token
    type: string
  from:
    title: Twilio phone number or the alphanumeric sender ID used
    type: string
  from_:
    duplicate: true
    title: Twilio phone number or the alphanumeric sender ID used
    type: string
  max_price:
    title: The total maximum price up to the fourth decimal (0.0001) in US dollars
      acceptable for the message to be delivered
    type: number
  media_url:
    format: uri
    title: The URL of the media you wish to send out with the message
    type: string
  message:
```

(continues on next page)

(continued from previous page)

```
    maxLength: 1600
    title: The text body of the message. Up to 1,600 characters long.
    type: string
  messaging_service_id:
    title: The unique id of the Messaging Service used with the message
    type: string
  provide_feedback:
    title: Set this value to true if you are sending messages that have a trackable
      user action and you intend to confirm delivery of the message using the Message
      Feedback API
    type: boolean
  status_callback:
    format: uri
    title: A URL where Twilio will POST each time your message status changes
    type: string
  to:
    title: The recipient of the message, in E.164 format
    format: e164
    type: string
  validity_period:
    maximum: 14400
    minimum: 1
    title: The number of seconds that the message can remain in a Twilio queue
    type: integer
type: object
```

7.1.16 Zulip

Send Zulip notifications

```
>>> from notifiers import get_notifier
>>> zulip = get_notifier('zulip')
>>> zulip.notify(message='Hi!', to='foo', email='foo@bar.com', api_key='KEY', domain=
↪'foobar')
```

Full schema:

```
additionalProperties: false
allOf:
- required:
  - message
  - email
  - api_key
  - to
- error_oneOf: Only one of 'domain' or 'server' is allowed
  oneOf:
  - required:
    - domain
  - required:
    - server
properties:
  api_key:
    title: User API Key
    type: string
  domain:
```

(continues on next page)

(continued from previous page)

```
    minLength: 1
    title: Zulip cloud domain
    type: string
  email:
    format: email
    title: User email
    type: string
  message:
    title: Message content
    type: string
  server:
    format: uri
    title: 'Zulip server URL. Example: https://myzulip.server.com'
    type: string
  subject:
    title: Title of the stream message. Required when using stream.
    type: string
  to:
    title: Target of the message
    type: string
  type:
    enum:
    - stream
    - private
    title: Type of message to send
    type: string
  type_:
    enum:
    - stream
    - private
    title: Type of message to send
    type: string
type: object
```

CHAPTER

EIGHT

API DOCUMENTATION

8.1 API Documentation

Internal API documentation

8.1.1 Core

Core API reference

class notifiers.core.**SchemaResource**

Base class that represent an object schema and its utility methods

_get_environs(*prefix: str = None*) → dict

Fetches set environment variables if such exist, via the *dict_from_environs()* Searches for *[PREFIX_NAME]_[PROVIDER_NAME]_[ARGUMENT]* for each of the arguments defined in the schema

Return type dict

Parameters **prefix** (str) – The environ prefix to use. If not supplied, uses the default

Returns A dict of arguments and value retrieved from environs

_merge_defaults(*data: dict*) → dict

Convenience method that calls *merge_dicts()* in order to merge default values

Return type dict

Parameters **data** (dict) – Notification data

Returns A merged dict of provided data with added defaults

_prepare_data(*data: dict*) → dict

Use this method to manipulate data that'll fit the respected provider API. For example, all provider must use the message argument but sometimes provider expects a different variable name for this, like text.

Return type dict

Parameters **data** (dict) – Notification data

Returns Returns manipulated data, if there's a need for such manipulations.

_process_data(****data*) → dict

The main method that process all resources data. Validates schema, gets environs, validates data, prepares it via provider requirements, merges defaults and check for data dependencies

Return type dict

Parameters **data** – The raw data passed by the notifiers client

Returns Processed data

abstract property _required
Will hold the schema's required part

abstract property _schema
Resource JSON schema without the required part

_validate_data(*data: dict*)
Validates data against provider schema. Raises *BadArguments* if relevant

Parameters **data** (dict) – Data to validate

Raises *BadArguments*

_validate_data_dependencies(*data: dict*) → dict
Validates specific dependencies based on the content of the data, as opposed to its structure which can be verified on the schema level

Return type dict

Parameters **data** (dict) – Data to validate

Returns Return data if its valid

Raises *NotifierException*

_validate_schema()
Validates provider schema for syntax issues. Raises *SchemaError* if relevant

Raises *SchemaError*

property arguments
Returns all of the provider argument as declared in the JSON schema

create_response(*data: dict = None*, *response: requests.models.Response = None*, *errors: list = None*) → notifiers.core.Response
Helper function to generate a *Response* object

Return type *Response*

Parameters

- **data** (dict) – The data that was used to send the notification
- **response** (Response) – requests.Response if exist
- **errors** (list) – List of errors if relevant

property defaults
A dict of default provider values if such is needed

abstract property name
Resource provider name

property required
Returns a dict of the relevant required parts of the schema

property schema
A property method that'll return the constructed provider schema. Schema MUST be an object and this method must be overridden

Returns JSON schema of the provider

class notifiers.core.**Provider**
The Base class all notification providers inherit from.

abstract _send_notification(*data: dict*) → notifiers.core.Response
The core method to trigger the provider notification. Must be overridden.

Return type *Response*

Parameters **data** (dict) – Notification data

property metadata
Returns a dict of the provider metadata as declared. Override if needed.

notify(*raise_on_errors: bool = False*, ***kwargs*) → notifiers.core.Response
The main method to send notifications. Prepares the data via the *_prepare_data()* method and then sends the notification via the *_send_notification()* method

Return type *Response*

Parameters

- **kwargs** – Notification data
- **raise_on_errors** (bool) – Should the *raise_on_errors()* be invoked immediately

Returns A *Response* object

Raises *NotificationError* if raise_on_errors is set to True and response contained errors

property resources
Return a list of names of relevant *ProviderResource* objects

class notifiers.core.**ProviderResource**
The base class that is used to fetch provider related resources like rooms, channels, users etc.

class notifiers.core.**Response**(*status: str*, *provider: str*, *data: dict*, *response: requests.models.Response = None*, *errors: list = None*)
A wrapper for the Notification response.

Parameters

- **status** – Response status string. SUCCESS or FAILED
- **provider** – Provider name that returned that response. Correlates to name
- **data** – The notification data that was used for the notification
- **response** – The response object that was returned. Usually requests.Response
- **errors** – Holds a list of errors if relevant

raise_on_errors()
Raises a *NotificationError* if response hold errors

Raises *NotificationError*: If response has errors

notifiers.core.**get_notifier**(*provider_name: str*, *strict: bool = False*) → notifiers.core.Provider
Convenience method to return an instantiated *Provider* object according to it name

Return type *Provider*

Parameters

- **provider_name** (str) – The name of the requested *Provider*
- **strict** (bool) – Raises a ValueError if the given provider string was not found

Returns *Provider* or None

Raises **ValueError** – In case `strict` is True and provider not found

`notifiers.core.`**all_providers**`()` → list
Returns a list of all *Provider* names

`notifiers.core.`**notify**(*provider_name: str*, ***kwargs*) → notifiers.core.Response
Quickly sends a notification without needing to get a notifier via the *get_notifier()* method.

Return type *Response*

Parameters

- **provider_name** (`str`) – Name of the notifier to use. Note that if this notifier name does not exist it will raise a
- **kwargs** – Notification data, dependant on provider

Returns *Response*

Raises *NoSuchNotifierError* If `provider_name` is unknown, will raise notification error

8.1.2 Logging

class `notifiers.logging.`**NotificationHandler**(*provider: str*, *defaults: dict = None*, ***kwargs*)
A `logging.Handler` that enables directly sending log messages to notifiers

emit(*record*)
Override the `emit()` method that takes the `msg` attribute from the log record passed

Parameters **record** – `logging.LogRecord`

handleError(*record*)
Handles any errors raised during the *emit()* method. Will only try to pass exceptions to fallback notifier (if defined) in case the exception is a sub-class of *NotifierException*

Parameters **record** – `logging.LogRecord`

init_providers(*provider*, *kwargs*)
Inits main and fallback provider if relevant

Parameters

- **provider** – Provider name to use
- **kwargs** – Additional kwargs

Raises **ValueError** – If provider name or fallback names are not valid providers, a `ValueError` will be raised

8.1.3 Providers

API documentation for the different providers.

class `notifiers.providers.email.`**SMTP**
Send emails via SMTP

base_url = None

property defaults
A dict of default provider values if such is needed

name = 'email'

site_url = 'https://en.wikipedia.org/wiki/Email'

class notifiers.providers.gitter.**Gitter**
Send Gitter notifications

message_url = '/{room_id}/chatMessages'

property metadata
Returns a dict of the provider metadata as declared. Override if needed.

site_url = 'https://gitter.im'

class notifiers.providers.gitter.**GitterMixin**
Shared attributes between *GitterRooms* and *Gitter*

base_url = 'https://api.gitter.im/v1/rooms'

name = 'gitter'

path_to_errors = ('errors', 'error')

class notifiers.providers.gitter.**GitterRooms**
Returns a list of Gitter rooms via token

resource_name = 'rooms'

class notifiers.providers.gmail.**Gmail**
Send email via Gmail

base_url = 'smtp.gmail.com'

property defaults
A dict of default provider values if such is needed

name = 'gmail'

site_url = 'https://www.google.com/gmail/about/'

class notifiers.providers.hipchat.**HipChat**
Send HipChat notifications

room_notification = '/{room}/notification'

site_url = 'https://www.hipchat.com/docs/apiv2'

user_message = '/{user}/message'

class notifiers.providers.hipchat.**HipChatMixin**
Shared attributed between resources and HipChatResourceProxy

base_url = 'https://{group}.hipchat.com'

name = 'hipchat'

path_to_errors = ('error', 'message')

rooms_url = '/v2/room'

users_url = '/v2/user'

class notifiers.providers.hipchat.**HipChatResourceMixin**
Common resources attributes that should not override *HipChat* attributes

class notifiers.providers.hipchat.**HipChatRooms**
Return a list of HipChat rooms

resource_name = 'rooms'

class notifiers.providers.hipchat.**HipChatUsers**
Return a list of HipChat users

resource_name = 'users'

class notifiers.providers.join.**Join**
Send Join notifications

property defaults
A dict of default provider values if such is needed

push_url = '/sendPush'

site_url = 'https://joaoapps.com/join/api/'

class notifiers.providers.join.**JoinDevices**
Return a list of Join devices IDs

devices_url = '/listDevices'

resource_name = 'devices'

class notifiers.providers.join.**JoinMixin**
Shared resources between *Join* and *JoinDevices*

base_url = 'https://joinjoaomgcd.appspot.com/_ah/api/messaging/v1'

name = 'join'

class notifiers.providers.pushbullet.**Pushbullet**
Send Pushbullet notifications

base_url = 'https://api.pushbullet.com/v2/pushes'

property defaults
A dict of default provider values if such is needed

site_url = 'https://www.pushbullet.com'

class notifiers.providers.pushbullet.**PushbulletDevices**
Return a list of Pushbullet devices associated to a token

devices_url = 'https://api.pushbullet.com/v2/devices'

resource_name = 'devices'

class notifiers.providers.pushbullet.**PushbulletMixin**
Shared attributes between *PushbulletDevices* and *Pushbullet*

name = 'pushbullet'

path_to_errors = ('error', 'message')

class notifiers.providers.pushover.**Pushover**
Send Pushover notifications

message_url = 'messages.json'

property metadata
Returns a dict of the provider metadata as declared. Override if needed.

name = 'pushover'

site_url = 'https://pushover.net/'

class notifiers.providers.pushover.**PushoverLimits**

    limits_url = 'apps/limits.json'

    resource_name = 'limits'

class notifiers.providers.pushover.**PushoverMixin**

    base_url = 'https://api.pushover.net/1/'

    name = 'pushover'

    path_to_errors = ('errors',)

class notifiers.providers.pushover.**PushoverResourceMixin**

class notifiers.providers.pushover.**PushoverSounds**

    resource_name = 'sounds'

    sounds_url = 'sounds.json'

class notifiers.providers.simplepush.**SimplePush**
Send SimplePush notifications

    base_url = 'https://api.simplepush.io/send'

    name = 'simplepush'

    site_url = 'https://simplepush.io/'

class notifiers.providers.slack.**Slack**
Send Slack webhook notifications

    base_url = 'https://hooks.slack.com/services/'

    name = 'slack'

    site_url = 'https://api.slack.com/incoming-webhooks'

class notifiers.providers.telegram.**Telegram**
Send Telegram notifications

    push_endpoint = '/sendMessage'

    site_url = 'https://core.telegram.org/'

class notifiers.providers.telegram.**TelegramMixin**
Shared resources between *TelegramUpdates* and *Telegram*

    base_url = 'https://api.telegram.org/bot{token}'

    name = 'telegram'

    path_to_errors = ('description',)

class notifiers.providers.telegram.**TelegramUpdates**
Return Telegram bot updates, correlating to the *getUpdates* method. Returns chat IDs needed to notifications

    resource_name = 'updates'

    updates_endpoint = '/getUpdates'

class notifiers.providers.zulip.**Zulip**
Send Zulip notifications

api_endpoint = '/api/v1/messages'

base_url = 'https://{domain}.zulipchat.com'

property defaults
A dict of default provider values if such is needed

name = 'zulip'

path_to_errors = ('msg',)

site_url = 'https://zulipchat.com/api/'

class notifiers.providers.twilio.**Twilio**
Send an SMS via a Twilio number

base_url = 'https://api.twilio.com/2010-04-01/Accounts/{}/Messages.json'

name = 'twilio'

path_to_errors = ('message',)

site_url = 'https://www.twilio.com/'

class notifiers.providers.pagerduty.**PagerDuty**
Send PagerDuty Events

base_url = 'https://events.pagerduty.com/v2/enqueue'

name = 'pagerduty'

path_to_errors = ('errors',)

site_url = 'https://v2.developer.pagerduty.com/'

class notifiers.providers.mailgun.**MailGun**
Send emails via MailGun

base_url = 'https://api.mailgun.net/v3/{domain}/messages'

name = 'mailgun'

path_to_errors = ('message',)

site_url = 'https://documentation.mailgun.com/'

class notifiers.providers.popcornnotify.**PopcornNotify**
Send PopcornNotify notifications

base_url = 'https://popcornnotify.com/notify'

name = 'popcornnotify'

path_to_errors = ('error',)

site_url = 'https://popcornnotify.com/'

class notifiers.providers.statuspage.**Statuspage**
Create Statuspage incidents

incidents_url = 'incidents.json'

realtime_statuses = ['investigating', 'identified', 'monitoring', 'resolved']

scheduled_statuses = ['scheduled', 'in_progress', 'verifying', 'completed']

class notifiers.providers.statuspage.**StatuspageComponents**
Return a list of Statuspage components for the page ID

components_url = 'components.json'

resource_name = 'components'

class notifiers.providers.statuspage.**StatuspageMixin**
Shared resources between *Statuspage* and *StatuspageComponents*

base_url = 'https://api.statuspage.io/v1//pages/{page_id}/'

name = 'statuspage'

path_to_errors = ('error',)

site_url = 'https://statuspage.io'

8.1.4 Exceptions

APi documentation of exceptions

exception notifiers.exceptions.**NotifierException**(**args*, ***kwargs*)
Base notifier exception. Catch this to catch all of `notifiers` errors

exception notifiers.exceptions.**BadArguments**(*validation_error: str*, **args*, ***kwargs*)
Raised on schema data validation issues

Parameters

- **validation_error** – The validation error message
- **args** – Exception arguments
- **kwargs** – Exception kwargs

exception notifiers.exceptions.**SchemaError**(*schema_error: str*, **args*, ***kwargs*)
Raised on schema issues, relevant probably when creating or changing a provider schema

Parameters

- **schema_error** – The schema error that was raised
- **args** – Exception arguments
- **kwargs** – Exception kwargs

exception notifiers.exceptions.**NotificationError**(**args*, ***kwargs*)
A notification error. Raised after an issue with the sent notification. Looks for `errors` key word in kwargs.

Parameters

- **args** – Exception arguments
- **kwargs** – Exception kwargs

exception notifiers.exceptions.**NoSuchNotifierError**(*name: str*, **args*, ***kwargs*)
An unknown notifier was requests, one that was not registered

8.1.5 Utils

Assorted helper utils

notifiers.utils.helpers.**text_to_bool**(*value: str*) → bool
Tries to convert a text value to a bool. If unsuccessful returns if value is None or not

Return type `bool`

Parameters **value** (str) – Value to check

notifiers.utils.helpers.**merge_dicts**(*target_dict: dict*, *merge_dict: dict*) → dict
Merges merge_dict into target_dict if the latter does not already contain a value for each of the key names in merge_dict. Used to cleanly merge default and environ data into notification payload.

Return type dict

Parameters

- **target_dict** (dict) – The target dict to merge into and return, the user provided data for example
- **merge_dict** (dict) – The data that should be merged into the target data

Returns A dict of merged data

notifiers.utils.helpers.**dict_from_environs**(*prefix: str*, *name: str*, *args: list*) → dict
Return a dict of environment variables correlating to the arguments list, main name and prefix like so: [prefix]_[name]_[arg]

Return type dict

Parameters

- **prefix** (str) – The environ prefix to use
- **name** (str) – Main part
- **args** (list) – List of args to iterate over

Returns A dict of found environ values

JSON schema related utils

notifiers.utils.schema.helpers.**one_or_more**(*schema: dict*, *unique_items: bool = True*, *min: int = 1*, *max: int = None*) → dict
Helper function to construct a schema that validates items matching *schema* or an array containing items matching *schema*.

Return type dict

Parameters

- **schema** (dict) – The schema to use
- **unique_items** (bool) – Flag if array items should be unique
- **min** (int) – Correlates to minLength attribute of JSON Schema array
- **max** (int) – Correlates to maxLength attribute of JSON Schema array

notifiers.utils.schema.helpers.**list_to_commas**(*list_of_args*) → str
Converts a list of items to a comma separated list. If list_of_args is not a list, just return it back

Return type str

Parameters **list_of_args** – List of items

Returns A string representing a comma separated list.

JSON schema custom formats

notifiers.utils.schema.formats.**is_iso8601**(*instance: str*)
Validates ISO8601 format

notifiers.utils.schema.formats.**is_rfc2822**(*instance: str*)
Validates RFC2822 format

notifiers.utils.schema.formats.**is_ascii**(*instance: str*)
Validates data is ASCII encodable

notifiers.utils.schema.formats.**is_valid_file**(*instance: str*)
Validates data is a valid file

notifiers.utils.schema.formats.**is_valid_port**(*instance: int*)
Validates data is a valid port

notifiers.utils.schema.formats.**is_timestamp**(*instance*)
Validates data is a timestamp

notifiers.utils.schema.formats.**is_e164**(*instance*)
Validates data is E.164 format

class notifiers.utils.requests.**RequestsHelper**
A wrapper around requests.Session which enables generically handling HTTP requests

classmethod request(*url: str*, *method: str*, *raise_for_status: bool = True*, *path_to_errors: tuple = None*, **args*, ***kwargs*) → tuple
A wrapper method for request`()`, which adds some defaults and logging

Return type tuple

Parameters

- **url** (str) – The URL to send the reply to
- **method** (str) – The method to use
- **raise_for_status** (bool) – Should an exception be raised for a failed response. Default is **True**
- **args** – Additional args to be sent to the request
- **kwargs** – Additional args to be sent to the request

Returns Dict of response body or original requests.Response

CHAPTER

NINE

DEVELOPMENT DOCUMENTATION

TBD

CHAPTER

TEN

DONATIONS

If you like this and want to buy me a cup of coffee, please click the donation button above or click this link

Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

n

INDEX

Symbols

A

B

C

D

E

G

H

I

J

L

M

N

O

P

R

S

T

U

Z