pushjack Documentation

Release 1.5.0

Derrick Gilland

Nov 27, 2018

Contents

Push notifications for APNS (iOS) and GCM (Android).

CHAPTER 1

Links

- Project: https://github.com/dgilland/pushjack
- Documentation: https://pushjack.readthedocs.io
- PyPi: https://pypi.python.org/pypi/pushjack/
- TravisCI: https://travis-ci.org/dgilland/pushjack

CHAPTER 2

Quickstart

Install using pip:

```
pip install pushjack
```

Whether using `APNS` or `GCM`, pushjack provides clients for each.

2.1 APNS

Send notifications using the `APNSClient` class:

```
from pushjack import APNSClient

client = APNSClient(certificate='<path/to/certificate.pem>',
                    default_error_timeout=10,
                    default_expiration_offset=2592000,
                    default_batch_size=100,
                    default_retries=5)

token = '<device token>'
alert = 'Hello world.'

# Send to single device.
# NOTE: Keyword arguments are optional.
res = client.send(token,
                  alert,
                  badge='badge count',
                  sound='sound to play',
                  category='category',
                  content_available=True,
                  title='Title',
                  title_loc_key='t_loc_key',
                  title_loc_args='t_loc_args',
```

(continues on next page)

(continued from previous page)

```
                  action_loc_key='a_loc_key',
                  loc_key='loc_key',
                  launch_image='path/to/image.jpg',
                  extra={'custom': 'data'})

# Send to multiple devices by passing a list of tokens.
client.send([token], alert, **options)
```

Access response data.

```
# List of all tokens sent.
res.tokens

# List of errors as APNSServerError objects
res.errors

# Dict mapping errors as token => APNSServerError object.
res.token_errors
```

Override defaults for error_timeout, expiration_offset, and batch_size.

```
client.send(token,
            alert,
            expiration=int(time.time() + 604800),
            error_timeout=5,
            batch_size=200)
```

Send a low priority message.

```
# The default is low_priority == False
client.send(token, alert, low_priority=True)
```

Get expired tokens.

```
expired_tokens = client.get_expired_tokens()
```

Close APNS connection.

```
client.close()
```

For the APNS sandbox, use `APNSSandboxClient` instead:

```
from pushjack import APNSSandboxClient
```

2.2 GCM

Send notifications using the `GCMClient` class:

```
from pushjack import GCMClient

client = GCMClient(api_key='<api-key>')

registration_id = '<registration id>'
alert = 'Hello world.'
```

(continues on next page)

(continued from previous page)

```
notification = {'title': 'Title', 'body': 'Body', 'icon': 'icon'}

# Send to single device.
# NOTE: Keyword arguments are optional.
res = client.send(registration_id,
                  alert,
                  notification=notification,
                  collapse_key='collapse_key',
                  delay_while_idle=True,
                  time_to_live=604800)

# Send to multiple devices by passing a list of ids.
client.send([registration_id], alert, **options)
```

Alert can also be be a dictionary with data fields.

```
alert = {'message': 'Hello world', 'custom_field': 'Custom Data'}
```

Alert can also contain the notification payload.

```
alert = {'message': 'Hello world', 'notification': notification}
```

Send a low priority message.

```
# The default is low_priority == False
client.send(registration_id, alert, low_priority=True)
```

Access response data.

```
# List of requests.Response objects from GCM Server.
res.responses

# List of messages sent.
res.messages

# List of registration ids sent.
res.registration_ids

# List of server response data from GCM.
res.data

# List of successful registration ids.
res.successes

# List of failed registration ids.
res.failures

# List of exceptions.
res.errors

# List of canonical ids (registration ids that have changed).
res.canonical_ids
```

For more details, please see the full documentation at https://pushjack.readthedocs.io.

CHAPTER 3

Guide

3.1 Installation

pushjack requires Python >= 2.6 or >= 3.3.

To install from PyPi:

```
pip install pushjack
```

3.2 Upgrading

3.2.1 From v0.5.0 to v1.0.0

There were several, major breaking changes in `v1.0.0`:

- Make APNS always return `APNSResponse` object instead of only raising `APNSSendError` when errors encountered. (**breaking change**)
- Remove APNS/GCM module send functions and only support client interfaces. (**breaking change**)
- Remove `config` argument from `APNSClient` and use individual function parameters as mapped below instead: (**breaking change**)
  - `APNS_ERROR_TIMEOUT` => `default_error_timeout`
  - `APNS_DEFAULT_EXPIRATION_OFFSET` => `default_expiration_offset`
  - `APNS_DEFAULT_BATCH_SIZE` => `default_batch_size`
- Remove `config` argument from `GCMClient` and use individual functionm parameters as mapped below instead: (**breaking change**)
  - `GCM_API_KEY` => `api_key`
- Remove `pushjack.clients` module. (**breaking change**)

- Remove `pushjack.config` module. (**breaking change**)
- Rename `GCMResponse.payloads` to `GCMResponse.messages`. (**breaking change**)

The motiviation behind these drastic changes were to eliminate multiple methods for sending tokens (removing module functions in favor of using client classes) and to simplify the overall implementation (eliminating a separate configuration module/implementation and instead passing config parameters directly into client class). This has lead to a smaller, easier to maintain codebase with fewer implementation details.

The module send functions are no longer implemented:

```
# This no longer works on v1.0.0.
from pushjack import apns, gcm

apns.send(...)
gcm.send(...)
```

Instead, the respective client classes must be used instead:

```
# This works on v1.0.0.
from pushjack import APNSClient, APNSSandboxClient, GCMClient

apns = APNSClient(...)
apns.send(...)

apns_sandbox = APNSSandboxClient(...)
apns_sandbox.send(...)

gcm = GCMClient(...)
gcm.send(...)
```

The configuration module has been eliminated:

```
# This fails on v1.0.0.
from pushjack import APNSClient, GCMClient, create_apns_config, create_gcm_config

apns = APNSClient(create_apns_config({
    'APNS_CERTIFICATE': '<path/to/certificate.pem>',
    'APNS_ERROR_TIMEOUT': 10,
    'APNS_DEFAULT_EXPIRATION_OFFSET: 60 * 60 * 24 * 30,
    'APNS_DEFAULT_BATCH_SIZE': 100
}))
apns.send(tokens, alert, **options)

gcm = GCMClient(create_gcm_config({
    'GCM_API_KEY': '<api-key>'
}))
gcm.send(tokens, alert, **options)
```

Instead, configuration values are passed directly during class instance creation:

```
# This works on v1.0.0.
from pushjack import APNSClient, APNSSandboxClient, GCMClient

apns = APNSClient('<path/to/certificate.pem>',
                  default_error_timeout=10,
                  default_expiration_offset=60 * 60 * 24 * 30,
                  default_batch_size=100)
```

(continues on next page)

(continued from previous page)

```
# or if wanting to use the sandbox:
# sandbox = APNSSandboxClient(...)

apns.send(tokens, alert, **options)

gcm = GCMClient('<api-key>')
gcm.send(tokens, alert, **options)
```

APNS sending no longer raises an `APNSSendError` when error encountered:

```
# This fails on v1.0.0
from pushjack APNSSendError

try:
    apns.send(tokens, alert, **options)
except APNSSendError as ex:
    ex.errors
```

Instead, APNS sending returns an *pushjack.apns.APNSResponse* object:

```
# This works on v1.0.0
res = apns.send(tokens, alert, **options)
res.errors
res.error_tokens
```

3.2.2 From v0.4.0 to v0.5.0

There were two breaking changes in `v0.5.0`:

- Make APNS `send` raise an `APNSSendError` when one or more error responses received. `APNSSendError` contains an aggregation of errors, all tokens attempted, failed tokens, and successful tokens. (**breaking change**)
- Replace `priority` argument to APNS `send` with `low_priority=False`. (**breaking change**)

The new exception `APNSSendError` replaces individually raised APNS server errors. So instead of catching the base server exception, `APNSServerError`, catch `APNSSendError` instead:

```
from pushjack import apns

# On v0.4.0
try:
    apns.send(tokens, **options)
except APNSServerError:
    pass

# Updated for v0.5.0
try:
    apns.send(tokens, **options)
except APNSSendError:
    pass
```

The new `low_priority` argument makes setting the APNS notification priority more straight-forward:

```
from pushjack import apns

# On v0.4.0
```

(continues on next page)

(continued from previous page)

```
## High priority (the default)
apns.send(tokens, alert)
apns.send(tokens, alert, priority=10)

## Low priority
apns.send(tokens, alert, priority=5)

# Updated for v0.5.0

## High priority (the default)
apns.send(tokens, alert)
apns.send(tokens, alert, low_priority=False)

## Low priority
apns.send(tokens, alert, low_priority=True)
```

3.2.3 From v0.3.0 to v0.4.0

There were several breaking changes in `v0.4.0`:

- Remove `request` argument from GCM send function. (**breaking change**)
- Remove `sock` argument from APNS send function. (**breaking change**)
- Remove APNS and GCM `send_bulk` function. Modify `send` to support bulk notifications. (**breaking change**)

The first two items should be fairly minor as these arguments were not well documented nor encouraged. In `v0.4.0` the APNS socket and GCM request objects are now managed within the send functions.

The last item is more likely to break code since `send_bulk` was removed. However, replacing `send_bulk` with `send` will fix it:

```
from pushjack import apns, gcm

# On v0.3.0
apns.send_bulk(tokens, **options)
gcm.send_bulk(tokens, **options)

# Updated for v0.4.0
apns.send(tokens, **options)
gcm.send(tokens, **options)
```

3.3 API Reference

3.3.1 APNS

Client module for Apple Push Notification service.

The algorithm used to send bulk push notifications is optimized to eagerly check for errors using a single thread. Error checking is performed after each batch send (bulk notifications may be broken up into multiple batches) and is non-blocking until the last notification is sent. A final, blocking error check is performed using a customizable error

timeout. This style of error checking is done to ensure that no errors are missed (e.g. by waiting too long to check errors before the connection is closed by the APNS server) without having to use two threads to read and write.

The return from a send operation will contain a response object that includes any errors encountered while sending. These errors will be associated with the failed tokens.

For more details regarding Apple's APNS documentation, consult the following:

- Apple Push Notification Service
- Provider Communication with APNS

class pushjack.apns.**APNSClient**(*certificate*, *default_error_timeout=10*, *default_expiration_offset=2592000*, *default_batch_size=100*, *default_max_payload_length=0*, *default_retries=5*)

APNS client class.

close()
Close APNS connection.

conn
Reference to lazy APNS connection.

create_connection()
Create and return new APNS connection to push server.

create_feedback_connection()
Create and return new APNS connection to feedback server.

get_expired_tokens()
Return inactive device tokens that are no longer registered to receive notifications.

Returns List of *APNSExpiredToken* instances.

Return type list

New in version 0.0.1.

send(*ids*, *message=None*, *expiration=None*, *low_priority=None*, *batch_size=None*, *error_timeout=None*, *max_payload_length=None*, *retries=None*, ***options*)
Send push notification to single or multiple recipients.

Parameters

- **ids** (*list*) – APNS device tokens. Each item is expected to be a hex string.
- **message** (*str|dict*) – Message string or APS dictionary. Set to `None` to send an empty alert notification.
- **expiration** (*int, optional*) – Expiration time of message in seconds offset from now. Defaults to `None` which uses `default_expiration_offset`.
- **low_priority** (*boolean, optional*) – Whether to send notification with the low priority flag. Defaults to `False`.
- **batch_size** (*int, optional*) – Number of notifications to group together when sending. Defaults to `None` which uses attr:*default_batch_size*.
- **error_timeout** (*int, optional*) – Time in seconds to wait for the error response after sending messages. Defaults to `None` which uses attr:*default_error_timeout*.
- **max_payload_length** (*int, optional*) – The maximum length of the payload to send. Message will be trimmed if the size is exceeded. Use 0 to turn off. Defaults to `None` which uses attr:*default_max_payload_length*.

- **retries** (*int, optional*) – Number of times to retry when the send operation fails. Defaults to `None` which uses `default_retries`.

Keyword Arguments

- **badge** (*int, optional*) – Badge number count for alert. Defaults to `None`.
- **sound** (*str, optional*) – Name of the sound file to play for alert. Defaults to `None`.
- **category** (*str, optional*) – Name of category. Defaults to `None`.
- **content_available** (*bool, optional*) – If `True`, indicate that new content is available. Defaults to `None`.
- **title** (*str, optional*) – Alert title.
- **title_loc_key** (*str, optional*) – The key to a title string in the `Localizable.strings` file for the current localization.
- **title_loc_args** (*list, optional*) – List of string values to appear in place of the format specifiers in *title_loc_key*.
- **action_loc_key** (*str, optional*) – Display an alert that includes the `Close` and `View` buttons. The string is used as a key to get a localized string in the current localization to use for the right button's title instead of `"View"`.
- **loc_key** (*str, optional*) – A key to an alert-message string in a `Localizable.strings` file for the current localization.
- **loc_args** (*list, optional*) – List of string values to appear in place of the format specifiers in `loc_key`.
- **launch_image** (*str, optional*) – The filename of an image file in the app bundle; it may include the extension or omit it.
- **mutable_content** (*bool, optional*) – if `True`, triggers Apple Notification Service Extension. Defaults to `None`.
- **thread_id** (*str, optional*) – Identifier for grouping notifications. iOS groups notifications with the same thread identifier together in Notification Center. Defaults to `None`.
- **extra** (*dict, optional*) – Extra data to include with the alert.

Returns

Response from APNS containing tokens sent and any errors encountered.

Return type *APNSResponse*

Raises

- `APNSInvalidTokenError` – Invalid token format. *APNSInvalidTokenError*
- `APNSInvalidPayloadSizeError` – Notification payload size too large. *APNSInvalidPayloadSizeError*
- `APNSMissingPayloadError` – Notificationpayload is empty. *APNSMissingPayloadError*

New in version 0.0.1.

Changed in version 0.4.0: - Added support for bulk sending. - Made sending and error checking non-blocking. - Removed *sock*, *payload*, and *identifer* arguments.

Changed in version 0.5.0: - Added `batch_size` argument. - Added `error_timeout` argument. - Replaced `priority` argument with `low_priority=False`. - Resume sending notifications when a sent token has an error response. - Raise `APNSSendError` if any tokens have an error response.

Changed in version 1.0.0: - Return *APNSResponse* instead of raising `APNSSendError`. - Raise *APNSMissingPayloadError* if payload is empty.

Changed in version 1.4.0: Added `retries` argument.

class pushjack.apns.**APNSSandboxClient**(*certificate*, *default_error_timeout=10*, *default_expiration_offset=2592000*, *default_batch_size=100*, *default_max_payload_length=0*, *default_retries=5*)

APNS client class for sandbox server.

class pushjack.apns.**APNSResponse**(*tokens*, *message*, *errors*)

Response from APNS after sending tokens.

tokens
list – List of all tokens sent during bulk sending.

message
APNSMessage – `APNSMessage` object sent.

errors
list – List of APNS exceptions for each failed token.

failures
list – List of all failed tokens.

successes
list – List of all successful tokens.

token_errors
dict – Dict mapping the failed tokens to their respective APNS exception.

New in version 1.0.0.

class pushjack.apns.**APNSExpiredToken**

Represents an expired APNS token with the timestamp of when it expired.

token
str – Expired APNS token.

timestamp
int – Epoch timestamp.

Exceptions

The *APNSServerError* class of exceptions represent error responses from APNS. These exceptions will contain attributes for `code`, `description`, and `identifier`. The `identifier` attribute is the list index of the token that failed. However, none of these exceptions will be raised directly. Instead, APNS server errors are collected and packaged into a *APNSResponse* object and returned by *APNSClient.send()*. This object provides a list of the raw exceptions as well as a mapping of the actual token and its associated error.

Below is a listing of APNS Server exceptions:

Exception	Code	Description
APNSProcessingError	1	Processing error
APNSMissingTokenError	2	Missing token
APNSMissingTopicError	3	Missing topic
APNSMissingPayloadError	4	Missing payload
APNSInvalidTokenSizeError	5	Invalid token size
APNSInvalidTopicSizeError	6	Invalid topic size
APNSInvalidPayloadSizeError	7	Invalid payload size
APNSInvalidTokenError	8	Invalid token
APNSShutdownError	10	Shutdown
APNSUnknownError	255	Unknown

class pushjack.exceptions.**APNSError**
Base exception for APNS errors.

class pushjack.exceptions.**APNSAuthError**
Exception with APNS certificate.

class pushjack.exceptions.**APNSServerError**(*identifier*)
Base exception for APNS Server errors.

class pushjack.exceptions.**APNSProcessingError**(*identifier*)
Exception for APNS processing error.

class pushjack.exceptions.**APNSMissingTokenError**(*identifier*)
Exception for APNS missing token error.

class pushjack.exceptions.**APNSMissingTopicError**(*identifier*)
Exception for APNS missing topic error.

class pushjack.exceptions.**APNSMissingPayloadError**(*identifier*)
Exception for APNS payload error.

class pushjack.exceptions.**APNSInvalidTokenSizeError**(*identifier*)
Exception for APNS invalid token size error.

class pushjack.exceptions.**APNSInvalidTopicSizeError**(*identifier*)
Exception for APNS invalid topic size error.

class pushjack.exceptions.**APNSInvalidPayloadSizeError**(*identifier*)
Exception for APNS invalid payload size error.

class pushjack.exceptions.**APNSInvalidTokenError**(*identifier*)
Exception for APNS invalid token error.

class pushjack.exceptions.**APNSShutdownError**(*identifier*)
Exception for APNS shutdown error.

class pushjack.exceptions.**APNSUnknownError**(*identifier*)
Exception for APNS unknown error.

3.3.2 GCM

Client module for Google Cloud Messaging service.

By default, sending notifications is optimized to deliver notifications to the maximum number of allowable recipients per HTTP request (currently 1,000 recipients as specified in the GCM documentation).

The return from a send operation will contain a response object that parses all GCM HTTP responses and groups them by errors, successful registration ids, failed registration ids, canonical ids, and the raw responses from each request.

For more details regarding Google's GCM documentation, consult the following:

- GCM for Android
- GCM Server Reference

class pushjack.gcm.**GCMClient**(*api_key*)
GCM client class.

conn
Reference to lazy GCM connection.

create_connection()
Create and return new GCM connection.

send(*ids*, *message*, ***options*)
Send push notification to single or multiple recipients.

Parameters

- **ids** (*list*) – GCM device registration IDs.
- **message** (*str|dict*) – Message string or dictionary. If `message` is a dict and contains the field `notification`, then it will be used for the `notification` payload.

Keyword Arguments

- **notificatoin** (*dict, optional*) – Notification payload. Can include the fields `body`, `title`, and `icon`.
- **collapse_key** (*str, optional*) – Identifier for a group of messages that can be collapsed so that only the last message gets sent when delivery can be resumed. Defaults to `None`.
- **delay_while_idle** (*bool, optional*) – If `True` indicates that the message should not be sent until the device becomes active.
- **time_to_live** (*int, optional*) – How long (in seconds) the message should be kept in GCM storage if the device is offline. The maximum time to live supported is 4 weeks. Defaults to `None` which uses the GCM default of 4 weeks.
- **low_priority** (*boolean, optional*) – Whether to send notification with the low priority flag. Defaults to `False`.
- **restricted_package_name** (*str, optional*) – Package name of the application where the registration IDs must match in order to receive the message. Defaults to `None`.
- **dry_run** (*bool, optional*) – If `True` no message will be sent but request will be tested.

Returns Response from GCM server.

Return type *GCMResponse*

Raises `GCMAuthError` – If `api_key` not set. *GCMAuthError*

New in version 0.0.1.

Changed in version 0.4.0: - Added support for bulk sending. - Removed *request* argument.

Changed in version 1.2.0: - Added `low_priority` argument.

class pushjack.gcm.**GCMResponse**(*responses*)

GCM server response with results parsed into *responses*, *messages*, *registration_ids*, *data*, *successes*, *failures*, *errors*, and *canonical_ids*.

responses
list – List of `requests.Response` objects from each GCM request.

messages
list – List of message data sent in each GCM request.

registration_ids
list – Combined list of all recipient registration IDs.

data
list – List of each GCM server response data.

successes
list – List of registration IDs that were sent successfully.

failures
list – List of registration IDs that failed.

errors
list – List of exception objects correponding to the registration IDs that ere not sent successfully. See `pushjack.exceptions`.

canonical_ids
list – List of registration IDs that have been reassigned a new ID. Each element is an instance of *GCMCanonicalID*.

class pushjack.gcm.**GCMCanonicalID**

Represents a canonical ID returned by the GCM Server. This object indicates that a previously registered ID has changed to a new one.

old_id
str – Previously registered ID.

new_id
str – New registration ID that should replace *old_id*.

Exceptions

The *GCMServerError* class of exceptions are contained in *GCMResponse.errors*. Each exception contains attributes for `code`, `description`, and `identifier` (i.e. the registration ID that failed).

Below is a listing of GCM Server exceptions:

Exception	Code	Description
GCMMissingRegistrationError	MissingRegistration	Missing registration ID
GCMInvalidRegistrationError	InvalidRegistration	Invalid registration ID
GCMUnregisteredDeviceError	NotRegistered	Device not registered
GCMInvalidPackageNameError	InvalidPackageName	Invalid package name
GCMMismatchedSenderError	MismatchSenderId	Mismatched sender ID
GCMMessageTooBigError	MessageTooBig	Message too big
GCMInvalidDataKeyError	InvalidDataKey	Invalid data key
GCMInvalidTimeToLiveError	InvalidTtl	Invalid time to live
GCMTimeoutError	Unavailable	Timeout
GCMInternalServerError	InternalServerError	Internal server error
GCMDeviceMessageRateExceededError	DeviceMessageRateExceeded	Message rate exceeded

class pushjack.exceptions.**GCMError**
Base exception for GCM errors.

class pushjack.exceptions.**GCMAuthError**
Exception for error with GCM API key.

class pushjack.exceptions.**GCMServerError**(*identifier*)
Base exception for GCM Server errors.

class pushjack.exceptions.**GCMMissingRegistrationError**(*identifier*)
Exception for missing registration ID.

class pushjack.exceptions.**GCMInvalidRegistrationError**(*identifier*)
Exception for invalid registration ID

class pushjack.exceptions.**GCMUnregisteredDeviceError**(*identifier*)
Exception for unregistered device.

class pushjack.exceptions.**GCMInvalidPackageNameError**(*identifier*)
Exception for invalid package name.

class pushjack.exceptions.**GCMMismatchedSenderError**(*identifier*)
Exception for mismatched sender.

class pushjack.exceptions.**GCMMessageTooBigError**(*identifier*)
Exception for message too big.

class pushjack.exceptions.**GCMInvalidDataKeyError**(*identifier*)
Exception for invalid data key.

class pushjack.exceptions.**GCMInvalidTimeToLiveError**(*identifier*)
Exception for invalid time to live.

class pushjack.exceptions.**GCMTimeoutError**(*identifier*)
Exception for server timeout.

class pushjack.exceptions.**GCMInternalServerError**(*identifier*)
Exception for internal server error.

class pushjack.exceptions.**GCMDeviceMessageRateExceededError**(*identifier*)
Exception for device message rate exceeded.

3.3.3 Logging

Internal logging is handled with the logging module. The logger names used are:

- `pushjack`
- `pushjack.apns`
- `pushjack.gcm`

Enabling

To enable logging using an imperative approach:

```
import logging
import pushjack

logger = logging.getLogger('pushjack')
```

(continues on next page)

(continued from previous page)

```
logger.setLevel(logging.DEBUG)

formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
stream_handler = logging.StreamHandler()
stream_handler.setFormatter(formatter)

logger.addHandler(stream_handler)
```

To enable logging using a configuration approach:

```
import logging
import logging.config
import pushjack

logging.config.dictConfig({
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'console': {
            'class': 'logging.StreamHandler',
            'level': 'DEBUG'
        }
    },
    'loggers': {
        'pushjack': {
            'handlers': ['console']
        }
    }
})
```

For additional configuration options, you may wish to install logconfig:

```
pip install logconfig
```

```
import logconfig
import pushjack

logconfig.from_yaml('path/to/logconfig.yml')
```

CHAPTER 4

Project Info

4.1 License

The MIT License (MIT)

Copyright (c) 2015 Derrick Gilland

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

4.2 Versioning

This project follows Semantic Versioning with the following caveats:

- Only the public API (i.e. the objects imported into the `pushjack` module) will maintain backwards compatibility between MINOR version bumps.
- Objects within any other parts of the library are not guaranteed to not break between MINOR version bumps.

With that in mind, it is recommended to only use or import objects from the main module, `pushjack`.

4.3 Changelog

4.3.1 v1.5.0 (2018-07-29)

- gcm: Use FCM URL instead of deprecated GCM URL. Thanks Lukas Anzinger!

4.3.2 v1.4.1 (2018-06-18)

- apns: Remove restriction on token length due to incorrect assumption about tokens always being 64 characters long.

4.3.3 v1.4.0 (2017-11-09)

- apns: Add exceptions `APNSProtocolError` and `APNSTimeoutError`. Thanks Jakub Kleň!
- apns: Add retry mechanism to `APNSClient.send`. Thanks Jakub Kleň!
  - Add `default_retries` argument to `APNSClient` initialization. Defaults to `5`.
  - Add `retries` argument to `APNSClient.send`. By default will use `APNSClient.default_retries` unless explicitly passed in.
  - If unable to send after `retries`, an `APNSTimeoutError` will be raised.
- apns: Fix bug in bulk `APNSClient.send` that resulted in an off-by-one error for message identifier in returned errors. Thanks Jakub Kleň!
- apns: Add max payload truncation option to `APNSClient.send`. Thanks Jakub Kleň!
  - Add `default_max_payload_length` argument to `APNSClient` initialization. Defaults to `0` which disabled max payload length check.
  - Add `max_payload_length` argument to `APNSClient.send`. By default will use `APNSClient.default_max_payload_length` unless explicitly passed in.
  - When `max_payload_length` set, messages will be truncated to fit within the length restriction by trimming the "message" text and appending it with "…".

4.3.4 v1.3.0 (2017-03-11)

- apns: Optimize reading from APNS Feedback so that the number of bytes read are based on header and token lengths.
- apns: Explicitly close connection to APNS Feedback service after reading data.
- apns: Add support for `mutable-content` field (Apple Notification Service Extension) via `mutable_content` argument to `APNSClient.send()`. Thanks Ahmed Khedr!
- apns: Add support for `thread-id` field (group identifier in Notification Center) via `thread_id` argument to `APNSClient.send()`. Thanks Ahmed Khedr!

4.3.5 v1.2.1 (2015-12-14)

- apns: Fix implementation of empty APNS notifications and allow notifications with `{"aps": {}}` to be sent. Thanks Julius Seporaitis!

4.3.6 v1.2.0 (2015-12-04)

- gcm: Add support for `priority` field to GCM messages via `low_priority` keyword argument. Default behavior is for all messages to be `"high"` priority. This is the opposite of GCM messages but mirrors the behavior in the APNS module where the default priority is `"high"`.

4.3.7 v1.1.0 (2015-10-22)

- gcm: Add support for `notification` field to GCM messages.
- gcm: Replace `registration_ids` field with `to` field when sending to a single recipient since `registration_ids` field has been deprecated for single recipients.

4.3.8 v1.0.1 (2015-05-07)

- gcm: Fix incorrect authorization header in GCM client. Thanks Brad Montgomery!

4.3.9 v1.0.0 (2015-04-28)

- apns: Add `APNSSandboxClient` for sending notifications to APNS sandbox server.
- apns: Add `message` attribute to `APNSResponse`.
- pushjack: Add internal logging.
- apns: Fix APNS error checking to properly handle reading when no data returned.
- apns: Make APNS sending stop during iteration if a fatal error is received from APNS server (e.g. invalid topic, invalid payload size, etc).
- apns/gcm: Make APNS and GCM clients maintain an active connection to server.
- apns: Make APNS always return `APNSResponse` object instead of only raising `APNSSendError` when errors encountered. (**breaking change**)
- apns/gcm: Remove APNS/GCM module send functions and only support client interfaces. (**breaking change**)
- apns: Remove `config` argument from `APNSClient` and use individual method parameters as mapped below instead: (**breaking change**)
  - `APNS_ERROR_TIMEOUT` => `default_error_timeout`
  - `APNS_DEFAULT_EXPIRATION_OFFSET` => `default_expiration_offset`
  - `APNS_DEFAULT_BATCH_SIZE` => `default_batch_size`
- gcm: Remove `config` argument from `GCMClient` and use individual method parameters as mapped below instead: (**breaking change**)
  - `GCM_API_KEY` => `api_key`
- pushjack: Remove `pushjack.clients` module. (**breaking change**)
- pushjack: Remove `pushjack.config` module. (**breaking change**)
- gcm: Rename `GCMResponse.payloads` to `GCMResponse.messages`. (**breaking change**)

4.3.10 v0.5.0 (2015-04-22)

- apns: Add new APNS configuration value `APNS_DEFAULT_BATCH_SIZE` and set to `100`.
- apns: Add `batch_size` parameter to APNS `send` that can be used to override `APNS_DEFAULT_BATCH_SIZE`.
- apns: Make APNS `send` batch multiple notifications into a single payload. Previously, individual socket writes were performed for each token. Now, socket writes are batched based on either the `APNS_DEFAULT_BATCH_SIZE` configuration value or the `batch_size` function argument value.
- apns: Make APNS `send` resume sending from after the failed token when an error response is received.
- apns: Make APNS `send` raise an `APNSSendError` when one or more error responses received. `APNSSendError` contains an aggregation of errors, all tokens attempted, failed tokens, and successful tokens. (**breaking change**)
- apns: Replace `priority` argument to APNS `send` with `low_priority=False`. (**breaking change**)

4.3.11 v0.4.0 (2015-04-15)

- apns: Improve error handling in APNS so that errors aren't missed.
- apns: Improve handling of APNS socket connection during bulk sending so that connection is re-established when lost.
- apns: Make APNS socket read/writes non-blocking.
- apns: Make APNS socket frame packing easier to grok.
- apns/gmc: Remove APNS and GCM `send_bulk` function. Modify `send` to support bulk notifications. (**breaking change**)
- apns: Remove `APNS_MAX_NOTIFICATION_SIZE` as config option.
- gcm: Remove `GCM_MAX_RECIPIENTS` as config option.
- gcm: Remove `request` argument from GCM send function. (**breaking change**)
- apns: Remove `sock` argument from APNS send function. (**breaking change**)
- gcm: Return namedtuple for GCM canonical ids.
- apns: Return namedtuple class for APNS expired tokens.

4.3.12 v0.3.0 (2015-04-01)

- gcm: Add `restricted_package_name` and `dry_run` fields to GCM sending.
- gcm: Add exceptions for all GCM server error responses.
- apns: Make `apns.get_expired_tokens` and `APNSClient.get_expired_tokens` accept an optional `sock` argument to provide a custom socket connection.
- apns: Raise `APNSAuthError` instead of `APNSError` if certificate file cannot be read.
- apns: Raise `APNSInvalidPayloadSizeError` instead of `APNSDataOverflow`. (**breaking change**)
- apns: Raise `APNSInvalidTokenError` instead of `APNSError`.
- gcm: Raise `GCMAuthError` if `GCM_API_KEY` is not set.
- pushjack: Rename several function parameters: (**breaking change**)

  - gcm: `alert` to `data`
  - gcm: `token`/`tokens` to `registration_id`/`registration_ids`
  - gcm: `Dispatcher`/`dispatcher` to `GCMRequest`/`request`
  - Clients: `registration_id` to `device_id`
- gcm: Return `GCMResponse` object for `GCMClient.send/send_bulk`. (**breaking change**)
- gcm: Return `requests.Response` object(s) for `gcm.send/send_bulk`. (**breaking change**)

4.3.13 v0.2.2 (2015-03-30)

- apns: Fix payload key assigments for `title-loc`, `title-loc-args`, and `launch-image`. Previously, `'_'` was used in place of `'-'`.

4.3.14 v0.2.1 (2015-03-28)

- apns: Fix incorrect variable reference in `apns.receive_feedback`.

4.3.15 v0.2.0 (2015-03-28)

- pushjack: Fix handling of `config` in clients when `config` is a class object and subclass of `Config`.
- apns: Make `apns.send/send_bulk` accept additional `alert` fields: `title`, `title-loc`, `title-loc-args`, and `launch-image`.
- gcm: Make `gcm.send/send_bulk` raise a `GCMError` exception if `GCM_API_KEY` is not set.
- gcm: Make gcm payload creation cast `data` to dict if isn't not passed in as one. Original value of `data` is then set to `{'message': data}`. (**breaking change**)
- gcm: Make gcm payload creation not set defaults for optional keyword arguments. (**breaking change**)

4.3.16 v0.1.0 (2015-03-26)

- pushjack: Rename `pushjack.settings` module to `pushjack.config`. (**breaking change**)
- apns/gcm: Allow config settings overrides to be passed into `create_gcm_config`, `create_apns_config`, and `create_apns_sandbox_config`.
- pushjack: Override `Config`'s `update()` method with custom method that functions similarly to `from_object()` except that it accepts a `dict` instead.

4.3.17 v0.0.1 (2015-03-25)

- First release.

4.4 Authors

4.4.1 Lead

- Derrick Gilland, dgilland@gmail.com, dgilland@github

4.4.2 Contributors

- Brad Montgomery, bradmontgomery@github
- Julius Seporaitis, seporaitis@github
- Ahmed Khedr, aakhedr@github
- Jakub Kleň, kukosk@github
- Lukas Anzinger, Lukas0907@github

4.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.5.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/dgilland/pushjack.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" or "help wanted" is open to whoever wants to implement it.

Write Documentation

Pushjack could always use more documentation, whether as part of the official pushjack docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/dgilland/pushjack.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.5.2 Get Started!

Ready to contribute? Here's how to set up `pushjack` for local development.

1. Fork the `pushjack` repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pushjack.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenv installed, this is how you set up your fork for local development:

```
$ cd pushjack
$ pip install -r requirements.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass linting and all unit tests by testing with tox across all supported Python versions:

```
$ tox
```

6. Add yourself to `AUTHORS.rst`.
7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

4.5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. The pull request should work for all versions Python that this project supports. Check https://travis-ci.org/dgilland/pushjack/pull_requests and make sure that the all environments pass.

4.6 Kudos

This project started as a port of django-push-notifications with the goal of just separating the APNS and GCM modules from the Django related items. However, the implementation details, internals, and API interface have changed in `pushjack` and is no longer compatible with django-push-notifications. But a special thanks goes out to the author and contributors of django-push-notifications who unknowingly helped start this project along.

CHAPTER 5

Indices and Tables

- genindex
- modindex
- search

Python Module Index

p

Index