Federation Documentation

Release 0.15.0

Jason Robinson

Feb 21, 2018

Contents

Python library to abstract social web federation protocols like Diaspora.

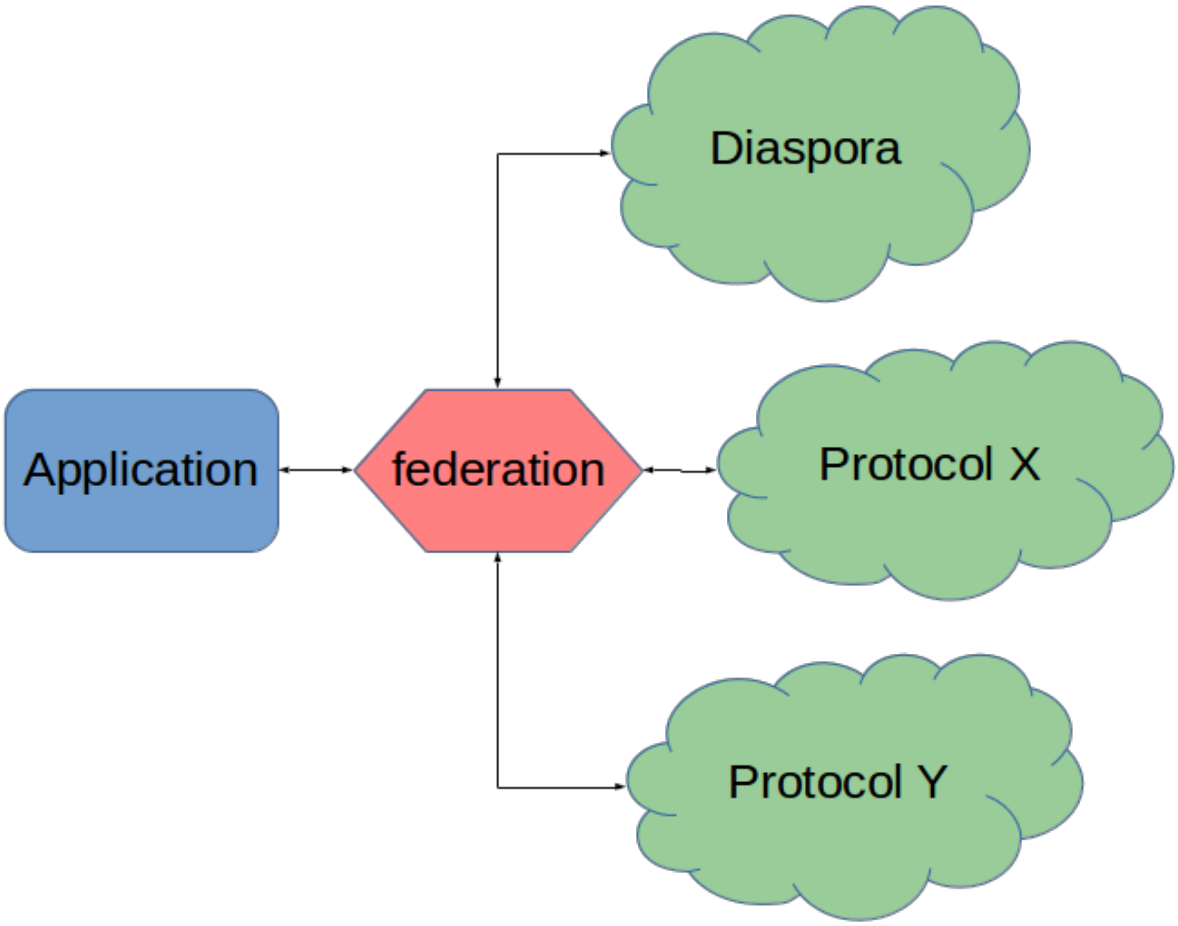
Contents:

CHAPTER 1

Introduction

The aim of *federation* is to provide and abstract multiple social web protocols like Diaspora in one package. This way applications can be built to (almost) transparently support many protocols without the app builder having to know everything about those protocols.

While the library does aim to provide an easy way to implement protocols like Diaspora into your application, it will not be a one to one mirror image of said protocols. The idea is to present one unified collection of entities and high level methods to the application to use. Since protocols can support different feature sets or have different ideas on even simple entities like status messages, it would be impossible to model the core entities according to a single protocol.



1.1 Status

Currently the library supports a part of the Diaspora protocol with remaining parts being constantly added. See the Diaspora protocol page for support status.

The code base is well tested and in use in several projects. Backward incompatible changes will however be made at this stage still, however those will be clearly documented in changelog entries.

1.2 Additional information

1.2.1 Installation and requirements

See installation documentation.

1.2.2 Usage and API documentation

See usage documentation.

1.2.3 Support and help

See development and support documentation.

1.2.4 License

BSD 3-clause license.

1.2.5 Author

Jason Robinson / jasonrobinson.me / GitHub

CHAPTER 2

Install

2.1 Dependencies

Depending on your operating system, certain dependencies will need to be installed.

2.1.1 lxml

lxml itself is installed by pip but the dependencies need to be installed as per lxml instructions.

2.2 Installation

Install with pip or include in your requirements file.

```
pip install federation
```

CHAPTER 3

Protocols

3.1 Diaspora

Currently the library supports a part of the protocol with remaining parts being constantly added.

Note! Diaspora project is currently rewriting parts of the protocol. This library aims to support the new version. When possible, compatibility will be kept with the current and legacy versions but this is not the main objective.

The feature set supported by this release is approximately the following:

- WebFinger, hCard and other discovery documents
- NodeInfo 1.0 documents
- Social-Relay documents
- Magic envelopes, signatures and other transport method related necessities
- **Entities as follows:**
  - Comment
  - Like
  - Photo
  - Profile
  - Retraction
  - StatusMessage
  - Contact
  - Reshare

Implementation unfortunately currently requires knowledge of how Diaspora discovery works as the implementer has to implement all the necessary views correctly (even though this library provides document generators). However, the magic envelope, signature and entity building is all abstracted inside the library.

For example implementations in real life projects check *Projects using federation*.

CHAPTER 4

Usage

4.1 Entities

Federation has it's own base entity classes. When incoming messages are processed, the protocol specific entity mappers transform the messages into our base entities. In reverse, when creating outgoing payloads, outgoing protocol specific messages are constructed from the base entities.

Entity types are as follows below.

class federation.entities.base.**Comment**(**args*, ***kwargs*)
Represents a comment, linked to another object.

class federation.entities.base.**Follow**(**args*, ***kwargs*)
Represents a handle following or unfollowing another handle.

class federation.entities.base.**Image**(**args*, ***kwargs*)
Reflects a single image, possibly linked to another object.

class federation.entities.base.**Post**(**args*, ***kwargs*)
Reflects a post, status message, etc, which will be composed from the message or to the message.

class federation.entities.base.**Profile**(**args*, ***kwargs*)
Represents a profile for a user.

class federation.entities.base.**Reaction**(**args*, ***kwargs*)
Represents a reaction to another object, for example a like.

class federation.entities.base.**Relationship**(**args*, ***kwargs*)
Represents a relationship between two handles.

class federation.entities.base.**Retraction**(**args*, ***kwargs*)
Represents a retraction of content by author.

class federation.entities.base.**Share**(**args*, ***kwargs*)
Represents a share of another entity.

`entity_type` defaults to "Post" but can be any base entity class name. It should be the class name of the entity that was shared.

The optional `raw_content` can be used for a "quoted share" case where the sharer adds their own note to the share.

4.1.1 Protocol entities

Each protocol additionally has it's own variants of the base entities, for example Diaspora entities in `federation.entities.diaspora.entities`. All the protocol specific entities subclass the base entities so you can safely work with for example `DiasporaPost` and use `isinstance(obj, Post)`.

When creating incoming objects from messages, protocol specific entity classes are returned. This is to ensure protocol specific extra attributes or methods are passed back to the caller.

For sending messages out, either base or protocol specific entities can be passed to the outbound senders. Base entities should be preferred unless the caller knows which protocol to send to.

If you need the correct protocol speficic entity class from the base entity, each protocol will define a `get_outbound_entity` function, for example the Diaspora function as follows.

`federation.entities.diaspora.mappers.`**`get_outbound_entity`**(*entity*, *private_key*)

Get the correct outbound entity for this protocol.

We might have to look at entity values to decide the correct outbound entity. If we cannot find one, we should raise as conversion cannot be guaranteed to the given protocol.

Private key of author is needed to be passed for signing the outbound entity.

Parameters **`entity`** – An entity instance which can be of a base or protocol entity class.

Returns Protocol specific entity class instance.

Raises **`ValueError`** – If conversion cannot be done.

4.2 Discovery

Federation provides many generators to allow providing the discovery documents that are necessary for the Diaspora protocol for example. The have been made as Pythonic as possible so that library users don't have to meddle with the various documents and their internals. Since each web framework will have it's own way of constructing views, one will still have to provide the view code to call the generators.

The protocols themselves are too complex to document within this library, please consult protocol documentation on what kind of discovery documents are expected to be served by the application.

4.2.1 Generators

Helper methods

`federation.hostmeta.generators.`**`generate_host_meta`**(*template=None*, **args*, ***kwargs*)

Generate a host-meta XRD document.

Template specific key-value pairs need to be passed as `kwargs`, see classes.

Parameters **`template`** – Ready template to fill with args, for example "diaspora" (optional)

Returns Rendered XRD document (str)

federation.hostmeta.generators.**generate_legacy_webfinger**(*template=None*, **args*, ***kwargs*)

Generate a legacy webfinger XRD document.

Template specific key-value pairs need to be passed as `kwargs`, see classes.

Parameters **template** – Ready template to fill with args, for example "diaspora" (optional)

Returns Rendered XRD document (str)

federation.hostmeta.generators.**generate_hcard**(*template=None*, ***kwargs*)

Generate a hCard document.

Template specific key-value pairs need to be passed as `kwargs`, see classes.

Parameters **template** – Ready template to fill with args, for example "diaspora" (optional)

Returns HTML document (str)

federation.hostmeta.generators.**get_nodeinfo_well_known_document**(*url*, *document_path=None*)

Generate a NodeInfo .well-known document.

See spec: http://nodeinfo.diaspora.software

Parameters

- **url** – The full base url with protocol, ie https://example.com
- **document_path** – Custom NodeInfo document path if supplied (optional)

Returns dict

Generator classes

class federation.hostmeta.generators.**DiasporaHostMeta**(**args*, ***kwargs*)

Diaspora host-meta.

Required keyword args:

- webfinger_host (str)

class federation.hostmeta.generators.**DiasporaWebFinger**(*handle*, *host*, *guid*, *public_key*, **args*, ***kwargs*)

Diaspora version of legacy WebFinger.

Required keyword args:

- handle (str) - eg user@domain.tld
- host (str) - eg https://domain.tld
- guid (str) - guid of user
- public_key (str) - public key

class federation.hostmeta.generators.**DiasporaHCard**(***kwargs*)

Diaspora hCard document.

Must receive the *required* attributes as keyword arguments to init.

class federation.hostmeta.generators.**NodeInfo**(*software*, *protocols*, *services*, *open_registrations*, *usage*, *metadata*, *skip_validate=False*, *raise_on_validate=False*)

Generate a NodeInfo document.

See spec: http://nodeinfo.diaspora.software

NodeInfo is unnecessarely restrictive in field values. We wont be supporting such strictness, though we will raise a warning unless validation is skipped with *skip_validate=True*.

For strictness, *raise_on_validate=True* will cause a *ValidationError* to be raised.

See schema document *federation/hostmeta/schemas/nodeinfo-1.0.json* for how to instantiate this class.

class federation.hostmeta.generators.**RFC3033Webfinger**(*id*, *base_url*, *profile_path*, *hcard_path='/hcard/users/'*, *atom_path=None*, *search_path=None*)

RFC 3033 webfinger - see https://diaspora.github.io/diaspora_federation/discovery/webfinger.html

A Django view is also available, see the child `django` module for view and url configuration.

Parameters

- **id** – Diaspora ID in URI format
- **base_url** – The base URL of the server (protocol://domain.tld)
- **profile_path** – Profile path for the user (for example */profile/johndoe/*)
- **hcard_path** – (Optional) hCard path, defaults to `/hcard/users/`.
- **atom_path** – (Optional) atom feed path

Returns dict

class federation.hostmeta.generators.**SocialRelayWellKnown**(*subscribe*, *tags=()*, *scope='all'*, **args*, ***kwargs*)

A *.well-known/social-relay* document in JSON.

For apps wanting to announce their preferences towards relay applications.

See WIP spec: https://wiki.diasporafoundation.org/Relay_servers_for_public_posts

Schema see *schemas/social-relay-well-known.json*

Parameters

- **subscribe** – bool
- **tags** – tuple, optional
- **scope** – Should be either "all" or "tags", default is "all" if not given

4.3 Fetchers

High level utility functions to fetch remote objects. These should be favoured instead of protocol specific utility functions.

federation.fetchers.**retrieve_remote_content**(*id*, *sender_key_fetcher=None*)

Retrieve remote content and return an Entity object.

Currently, due to no other protocols supported, always use the Diaspora protocol.

Parameters

- **id** – ID of the remote entity.

- **sender_key_fetcher** – Function to use to fetch sender public key. If not given, network will be used to fetch the profile and the key. Function must take handle as only parameter and return a public key.

Returns Entity class instance or `None`

`federation.fetchers.`**retrieve_remote_profile**(*handle*)

High level retrieve profile method.

Retrieve the profile from a remote location, using either the given protocol or by checking each protocol until a user can be constructed from the remote documents.

Currently, due to no other protocols supported, always use the Diaspora protocol.

Parameters **handle** – The profile handle in format username@domain.tld

Returns `federation.entities.base.Profile` or `None`

4.4 Inbound

High level utility functions to pass incoming messages to. These should be favoured instead of protocol specific utility functions.

`federation.inbound.`**handle_receive**(*payload*, *user=None*, *sender_key_fetcher=None*, *skip_author_verification=False*)

Takes a payload and passes it to the correct protocol.

Returns a tuple of:

- sender handle
- protocol name
- list of entities

NOTE! The returned sender is NOT necessarily the *author* of the entity. By sender here we're talking about the sender of the *payload*. If this object is being relayed by the sender, the author could actually be a different identity.

Parameters

- **payload** – Payload blob (str)
- **user** – User that will be passed to *protocol.receive* (only required on private encrypted content) MUST have a *private_key* and *guid* if given.
- **sender_key_fetcher** – Function that accepts sender handle and returns public key (optional)
- **skip_author_verification** – Don't verify sender (test purposes, false default)

Returns Tuple of sender handle, protocol name and list of entity objects

Raises **NoSuitableProtocolFound** – When no protocol was identified to pass message to

4.5 Outbound

High level utility functions to pass outbound entities to. These should be favoured instead of protocol specific utility functions.

federation.outbound.**handle_create_payload**(*entity*, *author_user*, *to_user_key=None*, *parent_user=None*)

Create a payload with the correct protocol.

Any given user arguments must have `private_key` and `handle` attributes.

Parameters

- **entity** – Entity object to send. Can be a base entity or a protocol specific one.
- **author_user** – User authoring the object.
- **to_user_key** – Public key of user private payload is being sent to, required for private payloads.
- **parent_user** – (Optional) User object of the parent object, if there is one. This must be given for the Diaspora protocol if a parent object exists, so that a proper `parent_author_signature` can be generated. If given, the payload will be sent as this user.

Returns Built payload message (str)

federation.outbound.**handle_send**(*entity*, *author_user*, *recipients=None*, *parent_user=None*)

Send an entity to remote servers.

Using this we will build a list of payloads per protocol, after resolving any that need to be guessed or looked up over the network. After that, each recipient will get the generated protocol payload delivered.

Any given user arguments must have `private_key` and `handle` attributes.

Parameters

- **entity** – Entity object to send. Can be a base entity or a protocol specific one.
- **author_user** – User authoring the object.
- **recipients** – A list of recipients to delivery to. Each recipient is a tuple containing at minimum the "id", optionally "public key" for private deliveries. Instead of a tuple, for public deliveries the "id" as str is also ok. If public key is provided, Diaspora protocol delivery will be made as an encrypted private delivery. For example [

  ("diaspora://user@domain.tld/profile/zyx", <RSAPublicKey object>), ("diaspora://user@domain2.tld/profile/xyz", None), "diaspora://user@domain3.tld/profile/xyz",

  ]
- **parent_user** – (Optional) User object of the parent object, if there is one. This must be given for the Diaspora protocol if a parent object exists, so that a proper `parent_author_signature` can be generated. If given, the payload will be sent as this user.

4.6 Django

Some ready provided views and URL configuration exist for Django.

Note! Django is not part of the normal requirements for this library. It must be installed separately.

4.6.1 Configuration

To use the Django views, ensure a modern version of Django is installed and add the views to your URL config for example as follows. The URL's must be mounted on root if Diaspora protocol support is required.

```
url(r"", include("federation.hostmeta.django.urls")),
```

Some settings need to be set in Django settings. An example is below:

```
FEDERATION = {
    "base_url": "https://myserver.domain.tld,
    "get_profile_function": "myproject.utils.get_profile_by_handle",
    "search_path": "/search/?q=",
}
```

- `base_url` is the base URL of the server, ie protocol://domain.tld.
- `profile_id_function` should be the full path to a function that given a handle will return a dictionary with information that will be used to generate the webfinger document. The dict should contain the following elements:
  - `id` - Diaspora URI format ID.
  - `profile_path` - profile path for generating an absolute URL to the profile page of the user.
  - `atom_path` - (optional) atom feed path for the profile
- `search_path` (optional) site search path which ends in a parameter for search input, for example "/search?q="

4.7 Protocols

The code for opening and creating protocol messages lives under each protocol module in `federation.protocols`. Currently Diaspora protocol is the only protocol supported.

Each protocol defines a `protocol.Protocol` class under it's own module. This is expected to contain certain methods that are used by the higher level functions that are called on incoming messages and when sending outbound messages. Everything that is needed to transform an entity into a message payload and vice versa should be here.

Instead of calling methods directly for a specific protocol, higher level generic functions should be normally used.

4.8 Utils

Various utils are provided for internal and external usage.

4.8.1 Diaspora

`federation.utils.diaspora.`**`fetch_public_key`**(*handle*)

Fetch public key over the network.

Parameters **`handle`** – Remote handle to retrieve public key for.

Returns Public key in str format from parsed profile.

federation.utils.diaspora.**generate_diaspora_profile_id**(*handle*, *guid=None*)
Generate a Diaspora profile ID from handle and guid.

Sometimes we don't know the guid if we just have a handle, but still we want to store it in URI format.

federation.utils.diaspora.**get_fetch_content_endpoint**(*domain*, *entity_type*, *guid*)
Get remote fetch content endpoint.

See: https://diaspora.github.io/diaspora_federation/federation/fetching.html

federation.utils.diaspora.**get_private_endpoint**(*id*)
Get remote endpoint for delivering private payloads.

federation.utils.diaspora.**get_public_endpoint**(*id*)
Get remote endpoint for delivering public payloads.

federation.utils.diaspora.**parse_diaspora_uri**(*uri*)
Parse Diaspora URI scheme string.

See: https://diaspora.github.io/diaspora_federation/federation/diaspora_scheme.html

Returns tuple of (handle, entity_type, guid) or `None`.

federation.utils.diaspora.**parse_profile_diaspora_id**(*id*)
Parse profile handle and guid from diaspora ID.

federation.utils.diaspora.**parse_profile_from_hcard**(*hcard*, *handle*)
Parse all the fields we can from a hCard document to get a Profile.

Parameters

- **hcard** – HTML hcard document (str)
- **handle** – User handle in username@domain.tld format

Returns `federation.entities.Profile` instance

federation.utils.diaspora.**retrieve_and_parse_content**(*id*, *sender_key_fetcher=None*)
Retrieve remote content and return an Entity class instance.

This is basically the inverse of receiving an entity. Instead, we fetch it, then call "handle_receive".

Parameters

- **id** – Diaspora URI scheme format ID.
- **sender_key_fetcher** – Function to use to fetch sender public key. If not given, network will be used to fetch the profile and the key. Function must take handle as only parameter and return a public key.

Returns Entity object instance or `None`

federation.utils.diaspora.**retrieve_and_parse_profile**(*handle*)
Retrieve the remote user and return a Profile object.

Parameters **handle** – User handle in username@domain.tld format

Returns `federation.entities.Profile` instance or None

federation.utils.diaspora.**retrieve_diaspora_hcard**(*handle*)
Retrieve a remote Diaspora hCard document.

Parameters **handle** – Remote handle to retrieve

Returns str (HTML document)

federation.utils.diaspora.**retrieve_diaspora_host_meta**(*host*)
Retrieve a remote Diaspora host-meta document.

Parameters **host** – Host to retrieve from

Returns XRD instance

4.8.2 Network

federation.utils.network.**fetch_document**(*url=None*, *host=None*, *path='/'*, *timeout=10*, *raise_ssl_errors=True*)
Helper method to fetch remote document.

Must be given either the `url` or `host`. If `url` is given, only that will be tried without falling back to http from https. If `host` given, *path* will be added to it. Will fall back to http on non-success status code.

Parameters

- **url** – Full url to fetch, including protocol
- **host** – Domain part only without path or protocol
- **path** – Path without domain (defaults to "/")
- **timeout** – Seconds to wait for response (defaults to 10)
- **raise_ssl_errors** – Pass False if you want to try HTTP even for sites with SSL errors (default True)

Returns Tuple of document (str or None), status code (int or None) and error (an exception class instance or None)

Raises **ValueError** – If neither url nor host are given as parameters

federation.utils.network.**send_document**(*url*, *data*, *timeout=10*, **args*, ***kwargs*)
Helper method to send a document via POST.

Additional `*args` and `**kwargs` will be passed on to `requests.post`.

Parameters

- **url** – Full url to send to, including protocol
- **data** – Dictionary (will be form-encoded), bytes, or file-like object to send in the body
- **timeout** – Seconds to wait for response (defaults to 10)

Returns Tuple of status code (int or None) and error (exception class instance or None)

4.9 Exceptions

Various custom exception classes might be returned.

exception federation.exceptions.**EncryptedMessageError**
Encrypted message could not be opened.

exception federation.exceptions.**NoSenderKeyFoundError**
Sender private key was not available to sign a payload message.

exception federation.exceptions.**NoSuitableProtocolFoundError**
No suitable protocol found to pass this payload message to.

exception federation.exceptions.**SignatureVerificationError**
Authenticity of the signature could not be verified given the key.

CHAPTER 5

Development

Help is more than welcome to extend this library. Please see the following resources.

- Source code repo
- Issue tracker
- Kanban board

5.1 Environment setup

Once you have your (Python 3.3+) virtualenv set up, install the development requirements:

```
pip install -r dev-requirements.txt
```

5.2 Running tests

```
py.test
```

5.3 Building local documentation

```
cd docs
make html
```

Built documentation is available at `docs/_build/html/index.html`.

5.4 Contact for help

Easiest via FreeNode IRC, channel `#python-federation` (webchat here).

There is also a Gitter chat available.

You can also ask questions or give feedback via issues.

CHAPTER 6

Projects using federation

For examples on how to integrate this library into your project, check these examples:

- Socialhome - a federated home page builder slash personal social network server with high emphasis on card style content visualization.
- Social-Relay - a reference server for the public content relay system that uses the Diaspora protocol.

CHAPTER 7

Changelog

7.1 [unreleased]

7.1.1 Added

- Enable generating encrypted JSON payloads with the Diaspora protocol which adds private message support. (related issue)

  JSON encrypted payload encryption and decryption is handled by the Diaspora `EncryptedPayload` class.

- Add RFC3033 webfinger generator (related issue)

  Also provided is a Django view and url configuration for easy addition into Django projects. Django is not a hard dependency of this library, usage of the Django view obviously requires installing Django itself. For configuration details see documentation.

7.1.2 Changed

- Send outbound Diaspora payloads in new format. Remove possibility to generate legacy MagicEnvelope payloads. (related issue)
- **Backwards incompatible**. Refactor `handle_send` function

  Now handle_send high level outbound helper function also allows delivering private payloads using the Diaspora protocol. (related issue)

  The signature has changed. Parameter `recipients` should now be a list of recipients to delivery to. Each recipient should either be an `id` or a tuple of `(id, public key)`. If public key is provided, Diaspora protocol delivery will be made as an encrypted private delivery.

- **Backwards incompatible**. Change `handle_create_payload` function signature.

  Parameter `to_user` is now `to_user_key` and thus instead of an object containing the `key` attribute it should now be an RSA public key object instance. This simplifies things since we only need the key from the user, nothing else.

7.2 [0.15.0] - 2018-02-12

7.2.1 Added

- Added base entity `Share` which maps to a `DiasporaReshare` for the Diaspora protocol. (related issue)

  The `Share` entity supports all the properties that a Diaspora reshare does. Additionally two other properties are supported: `raw_content` and `entity_type`. The former can be used for a "quoted share" case where the sharer adds their own note to the share. The latter can be used to reference the type of object that was shared, to help the receiver, if it is not sharing a `Post` entity. The value must be a base entity class name.

- Entities have two new properties: `id` and `target_id`.

  Diaspora entity ID's are in the form of the Diaspora URI scheme, where it is possible to construct an ID from the entity. In the future, ActivityPub object ID's will be found in these properties.

- New high level fetcher function `federation.fetchers.retrieve_remote_content`. (related issue)

  This function takes the following parameters:

  - `id` - Object ID. For Diaspora, the only supported protocol at the moment, this is in the Diaspora URI format.
  - `sender_key_fetcher` - Optional function that takes a profile `handle` and returns a public key in `str` format. If this is not given, the public key will be fetched from the remote profile over the network.

  The given ID will be fetched from the remote endpoint, validated to be from the correct author against their public key and then an instance of the entity class will be constructed and returned.

- New Diaspora protocol helpers in `federation.utils.diaspora`:
  - `retrieve_and_parse_content`. See notes regarding the high level fetcher above.
  - `fetch_public_key`. Given a `handle` as a parameter, will fetch the remote profile and return the `public_key` from it.
  - `parse_diaspora_uri`. Parses a Diaspora URI scheme string, returns either `None` if parsing fails or a `tuple` of `handle`, `entity_type` and `guid`.

- Support fetching new style Diaspora protocol Webfinger (RFC 3033) (related issue)

  The legaxy Webfinger is still used as fallback if the new Webfinger is not found.

7.2.2 Changed

- Refactoring for Diaspora `MagicEnvelope` class.

  The class init now also allows passing in parameters to construct and verify MagicEnvelope instances. The order of init parameters has not been changed, but they are now all optional. When creating a class instance, one should always pass in the necessary parameters depnding on whether the class instance will be used for building a payload or verifying an incoming payload. See class docstring for details.

- Diaspora procotol receive flow now uses the `MagicEnvelope` class to verify payloads. No functional changes regarding verification otherwise.

- Diaspora protocol receive flow now fetches the sender public key over the network if a `sender_key_fetcher` function is not passed in. Previously an error would be raised.

  Note that fetching over the network for each payload is wasteful. Implementers should instead cache public keys when possible and pass in a function to retrieve them, as before.

7.2.3 Fixed

- Converting base entity `Profile` to `DiasporaProfile` for outbound sending missed two attributes, `image_urls` and `tag_list`. Those are now included so that the values transfer into the built payload.
- Fix fallback to HTTP in the `fetch_document` network helper in the case of `ConnectionError` when trying HTTPS. Thanks @autogestion.
- Ensure `handle` is always lower cased when fetching remote profile using `retrieve_remote_profile`. Warning will be logged if an upper case handle is passed in.

7.3 [0.14.1] - 2017-08-06

7.3.1 Fixed

- Fix regression in handling Diaspora relayables due to security fix in 0.14.0. Payload and entity handle need to be allowed to be different when handling relayables.

7.4 [0.14.0] - 2017-08-06

7.4.1 Security

- Add proper checks to make sure Diaspora protocol payload handle and entity handle are the same. Even though we already verified the signature of the sender, we didn't ensure that the sender isn't trying to fake an entity authored by someone else.

  The Diaspora protocol functions `message_to_objects` and `element_to_objects` now require a new parameter, the payload sender handle. These functions should normally not be needed to be used directly.

7.4.2 Changed

- **Breaking change.** The high level `federation.outbound` functions `handle_send` and `handle_create_payload` signatures have been changed. This has been done to better represent the objects that are actually sent in and to add an optional `parent_user` object.

  For both functions the `from_user` parameter has been renamed to `author_user`. Optionally a `parent_user` object can also be passed in. Both the user objects must have `private_key` and `handle` attributes. In the case that `parent_user` is given, that user will be used to sign the payload and for Diaspora relayables an extra `parent_author_signature` in the payload itself.

7.5 [0.13.0] - 2017-07-22

7.5.1 Backwards incompatible changes

- When processing Diaspora payloads, entity used to get a `_source_object` stored to it. This was an `etree.Element` created from the source object. Due to serialization issues in applications (for example pushing the object to a task queue or saving to database), `_source_object` is now a byte string representation for the element done with `etree.tostring()`.

7.5.2 Added

- New style Diaspora private encrypted JSON payloads are now supported in the receiving side. Outbound private Diaspora payloads are still sent as legacy encrypted payloads. (issue)
  - No additional changes need to be made when calling `handle_receive` from your task processing. Just pass in the full received XML or JSON payload as a string with recipient user object as before.
- Add `created_at` to Diaspora `Comment` entity XML creator. This is required in renewed Diaspora protocol. (related issue)

7.5.3 Fixed

- Fix getting sender from a combination of legacy Diaspora encrypted payload and new entity names (for example `author`). This combination probably only existed in this library.
- Correctly extend entity `_children`. Certain Diaspora payloads caused `_children` for an entity to be written over by an empty list, causing for example status message photos to not be saved. Correctly do an extend on it. (issue)
- Fix parsing Diaspora profile `tag_string` into `Profile.tag_list` if the `tag_string` is an empty string. This caused the whole `Profile` object creation to fail. (issue)
- Fix processing Diaspora payload if it is passed to `handle_receive` as a `bytes` object. (issue)
- Fix broken Diaspora relayables after latest 0.2.0 protocol changes. Previously relayables worked only because they were reverse engineered from the legacy protocol. Now that XML order is not important and tag names can be different depending on which protocol version, the relayable forwarding broke. To fix, we don't regenerate the entity when forwarding it but store the original received object when generating a `parent_author_signature` (which is optional in some cases, but we generate it anyway for now). This happens in the previously existing `entity.sign_with_parent()` method. In the sending part, if the original received object (now with a parent author signature) exists in the entity, we send that to the remote instead of serializing the entity to XML.
  - To forward a relayable you must call `entity.sign_with_parent()` before calling `handle_send` to send the entity.

7.5.4 Removed

- `Post.photos` entity attribute was never used by any code and has been removed. Child entities of type `Image` are stored in the `Post._children` as before.
- Removed deprecated user private key lookup using `user.key` in Diaspora receive processing. Passed in `user` objects must now have a `private_key` attribute.

7.6 [0.12.0] - 2017-05-22

7.6.1 Backwards incompatible changes

- Removed exception class `NoHeaderInMessageError`. New style Diaspora protocol does not have a custom header in the Salmon magic envelope and thus there is no need to raise this anywhere.

7.6.2 Added

- New style Diaspora public payloads are now supported (see here). Old style payloads are still supported. Payloads are also still sent out old style.
- Add new `Follow` base entity and support for the new Diaspora "contact" payload. The simple `Follow` maps to Diaspora contact entity with following/sharing both true or false. Sharing as a separate concept is not currently supported.
- Added `_receiving_guid` to all entities. This is filled with `user.guid` if `user` is passed to `federation.inbound.handle_receive` and it has a `guid`. Normally in for example Diaspora, this will always be done in private payloads.

7.6.3 Fixed

- Legacy Diaspora retraction of sharing/following is now supported correctly. The end result is a `DiasporaRetraction` for entity type `Profile`. Since the payload doesn't contain the receiving user for a sharing/following retraction in legacy Diaspora protocol, we store the guid of the user in the entity as `_receiving_guid`, assuming it was passed in for processing.

7.7 [0.11.0] - 2017-05-08

7.7.1 Backwards incompatible changes

Diaspora protocol support added for `comment` and `like` relayable types. On inbound payloads the signature included in the payload will be verified against the sender public key. A failed verification will raise `SignatureVerificationError`. For outbound entities, the author private key will be used to add a signature to the payload.

This introduces some backwards incompatible changes to the way entities are processed. Diaspora entity mappers `get_outbound_entity` and entity utilities `get_full_xml_representation` now requires the author `private_key` as a parameter. This is required to sign outgoing `Comment` and `Reaction` (like) entities.

Additionally, Diaspora entity mappers `message_to_objects` and `element_to_objects` now take an optional `sender_key_fetcher` parameter. This must be a function that when called with the sender handle will return the sender public key. This allows using locally cached public keys instead of fetching them as needed. NOTE! If the function is not given, each processed payload will fetch the public key over the network.

A failed payload signature verification now raises a `SignatureVerificationError` instead of a less specific `AssertionError`.

7.7.2 Added

- Three new attributes added to entities.
  - Add protocol name to all entities to attribute `_source_protocol`. This might be useful for applications to know which protocol payload the entity was created from once multiple protocols are implemented.
  - Add source payload object to the entity at `_source_object` when processing it.
  - Add sender public key to the entity at `_sender_key`, but only if it was used for validating signatures.
- Add support for the new Diaspora payload properties coming in the next protocol version. Old XML payloads are and will be still supported.

- `DiasporaComment` and `DiasporaLike` will get the order of elements in the XML payload as a list in `xml_tags`. For implementers who want to recreate payloads for these relayables, this list should be saved for later use.
- High level `federation.outbound.handle_send` helper function now allows sending entities to a list of recipients without having to deal with payload creation or caring about the protocol (in preparation of being a multi-protocol library).
  - The function takes three parameters, `entity` that will be sent, `from_user` that is sending (note, not necessarely authoring, this user will be used to sign the payload for Diaspora for example) and a list of recipients as tuples of recipient handle/domain and optionally protocol. In the future, if protocol is not given, it will be guessed from the recipient handle, and if necessary a network lookup will be made to see what protocols the receiving identity supports.
  - Payloads will be delivered to each receiver only once. Currently only public messages are supported through this helper, so multiple recipients on a single domain will cause only one delivery.

7.7.3 Changed

- Refactor processing of Diaspora payload XML into entities. Diaspora protocol is dropping the `<XML><post></post></XML>` wrapper for the payloads. Payloads with the wrapper will still be parsed as before.

7.8 [0.10.1] - 2017-03-09

7.8.1 Fixes

- Ensure tags are lower cased after collecting them from entity `raw_content`.

7.9 [0.10.0] - 2017-01-28

7.9.1 Added

- Add support for new Diaspora protocol ISO 8601 timestamp format introduced in protocol version 0.1.6.
- Tests are now executed also against Python 3.6.

7.9.2 Fixes

- Don't crash `federation.utils.diaspora.retrieve_diaspora_webfinger` if XRD parse raises an `xml.parsers.expat.ExpatError`.

7.10 [0.9.1] - 2016-12-10

7.10.1 Fixes

- Made `Profile.raw_content` optional. This fixes validating profiles parsed from Diaspora hCard's.

7.11 [0.9.0] - 2016-12-10

7.11.1 Backwards incompatible changes

- `Image` no longer has a `text` attribute. It is replaced by `raw_content`, the same attribute as `Post` and `Comment` have. Unlike the latter two, `Image.raw_content` is not mandatory.

7.11.2 Added

- Entities can now have a children. These can be accessed using the `_children` list. Acceptable children depends on the entity. Currently, `Post`, `Comment` and `Profile` can have children of entity type `Image`. Child types are validated in the `.validate()` entity method call.

7.11.3 Fixed

- Diaspora protocol `message_to_objects` method (called through inbound high level methods) now correctly parses Diaspora `<photo>` elements and creates `Image` entities from them. If they are children of status messages, they will be available through the `Post._children` list.

7.12 [0.8.2] - 2016-10-23

7.12.1 Fixed

- Remove legacy splitting of payload to 60 chars when creating Diaspora payloads. Diaspora 0.6 doesn't understand these any more.

7.13 [0.8.1] - 2016-10-18

7.13.1 Fixed

- `federation.utils.network.send_document` incorrectly passed in `kwargs` to `requests.post`, causing an error when sending custom headers.
- Make sure `federation.utils.network.send_document` headers are treated case insensitive before passing then onwards to `requests.post`.

7.14 [0.8.0] - 2016-10-09

7.14.1 Library is now called `federation`

The name Social-Federation was really only an early project name which stuck. Since the beginning, the main module has been `federation`. It makes sense to unify these and also shorter names are generally nicer.

What do you need to do?

Mostly nothing since the module was already called `federation`. Some things to note below:

- Update your requirements with the new library name `federation`.
- If you hook to the old logger `social-federation`, update those to listen to `federation`, which is now the standard logger name used throughout.

7.14.2 Other backwards incompatible changes

- `federation.utils.diaspora.retrieve_and_parse_profile` will now return `None` if the `Profile` retrieved doesn't validate. This will affect also the output of `federation.fetchers.retrieve_remote_profile` which is the high level function to retrieve profiles.
- Remove unnecessary `protocol` parameter from `federation.fetchers.retrieve_remote_profile`. We're miles away from including other protocols and ideally the caller shouldn't have to pass in the protocol anyway.

7.14.3 Added

- Added `Retraction` entity with `DiasporaRetraction` counterpart.

7.15 [0.7.0] - 2016-09-15

7.15.1 Backwards incompatible changes

- Made `guid` mandatory for `Profile` entity. Library users should always be able to get a full validated object as we consider `guid` a core attribute of a profile.
- Always validate entities created through `federation.entities.diaspora.mappers.message_to_objects`. This is the code that transforms federation messages for the Diaspora protocol to actual entity objects. Previously no validation was done and callers of `federation.inbound.handle_receive` received entities that were not always valid, for example they were missing a `guid`. Now validation is done in the conversion stage and errors are pushed to the `federation` logger in the event of invalid messages.
  - Note Diaspora Profile XML messages do not provide a GUID. This is handled internally by fetching the guid from the remote hCard so that a valid `Profile` entity can be created.

7.15.2 Added

- Raise a warning if unknown parameters are passed to entities.
- Ensure entity required attributes are validated for `None` or empty string values. Required attributes must not only exist but also have a value.
- Add validation to entities with the attribute `public`. Only `bool` values are accepted.

7.15.3 Changed

- Function `federation.utils.diaspora.parse_profile_from_hcard` now requires a second argument, `handle`. Since in the future Diaspora hCard is not guaranteed to have username and domain, we now pass handle to the parser directly.

7.16 [0.6.1] - 2016-09-14

7.16.1 Fixed

- New style Diaspora Magic Envelope didn't require or like payload data to be cut to 60 char lines, as the legacy protocol does. Fixed to not cut lines.

7.17 [0.6.0] - 2016-09-13

7.17.1 Added

- New style Diaspora Magic Envelope support. The magic envelope can be created using the class `federation.protocols.diaspora.magic_envelope.MagicEnvelope`. By default this will not wrap the payload message in `<XML><post></post></XML>`. To provide that functionality the class should be initialized with `wrap_payload=True`. No changes are made to the protocol send methods yet, if you need this new magic envelope you can initialize and render it directly.

7.17.2 Changed

- Deprecate receiving user `key` attribute for Diaspora protocol. Instead correct attribute is now `private_key` for any user passed to `federation.inbound.handle_receive`. We already use `private_key` in the message creation code so this is just to unify the user related required attributes.
  - DEPRECATION: There is a fallback with `key` for user objects in the receiving payload part of the Diaspora protocol until 0.8.0.

7.17.3 Fixes

- Loosen up hCard selectors when parsing profile from hCard document in `federation.utils.diaspora.parse_profile_from_hcard`. The selectors now match Diaspora upcoming federation documentation.

7.18 [0.5.0] - 2016-09-05

7.18.1 Breaking changes

- `federation.outbound.handle_create_payload` parameter `to_user` is now optional. Public posts don't need a recipient. This also affects Diaspora protocol `build_send` method where the change is reflected similarly. #43
  - In practise this means the signature has changed for `handle_create_payload` and `build_send` from **`from_user, to_user, entity`** to **`entity, from_user, to_user=None`**.

7.18.2 Added

- `Post.provider_display_name` is now supported in the entity outbound/inbound mappers. #44
- Add utility method `federation.utils.network.send_document` which is just a wrapper around `requests.post`. User agent will be added to the headers and exceptions will be silently captured and returned instead. #45
- Add Diaspora entity utility `federation.entities.diaspora.utils.get_full_xml_representation`. Renders the entity XML document and wraps it in `<XML><post>...</post></XML>`. #46

7.19 [0.4.1] - 2016-09-04

7.19.1 Fixes

- Don't quote/encode `Protocol.build_send` payload. It was doing it wrongly in the first place and also it's not necessary since Diaspora 0.6 protocol changes. #41
- Fix identification of Diaspora protocol messages. This was not working in the case that the attributes in the tag were in different order. #41

7.20 [0.4.0] - 2016-07-24

7.20.1 Breaking changes

- While in early stages, doing some renaming of modules to suit the longer term. `federation.controllers` has been split into two, `federation.outbound` and `federation.inbound`. The following methods have new import locations:
  - `federation.controllers.handle_receive` -> `federation.inbound.handle_receive`
  - `federation.controllers.handle_create_payload` -> `federation.outbound.handle_create_payload`
- Class `federation.hostmeta.generators.DiasporaHCard` now requires `guid`, `public_key` and `username` for initialization. Leaving these out was a mistake in the initial implementation. Diaspora has these in at least 0.6 development branch.

7.20.2 Added

- `Relationship` base entity which represents relationships between two handles. Types can be following, sharing, ignoring and blocking. The Diaspora counterpart, `DiasporaRequest`, which represents a sharing/following request is outwards a single entity, but incoming a double entity, handled by creating both a sharing and following version of the relationship.
- `Profile` base entity and Diaspora counterpart `DiasporaProfile`. Represents a user profile.
- `federation.utils.network.fetch_document` utility function to fetch a remote document. Returns document, status code and possible exception. Takes either `url` or a `host` + `path` combination. With `host`, https is first tried and optionally fall back to http.

- Utility methods to retrieve Diaspora user discovery related documents. These include the host-meta, webfinger and hCard documents. The utility methods are in `federation.utils.diaspora`.
- Utility to fetch remote profile, `federation.fetchers.retrieve_remote_profile`. Currently always uses Diaspora protocol. Returns a `Profile` entity.

7.20.3 Changed

- Unlock most of the direct dependencies to a certain version range. Unlock all of test requirements to any version.
- Entities passed to `federation.controllers.handle_create_payload` are now converted from the base entity types (Post, Comment, Reaction, etc) to Diaspora entity types (DiasporaPost, DiasporaComment, DiasporaLike, etc). This ensures actual payload generation has the correct methods available (for example `to_xml`) whatever entity is passed in.

7.20.4 Fixes

- Fix fetching sender handle from Diaspora protocol private messages. As it is not contained in the header, it needs to be read from the message content itself.
- Fix various issues with `DiasporaHCard` template after comparing to some real world hCard templates from real pods. Old version was based on documentation in Diaspora project wiki.

7.21 [0.3.2] - 2016-05-09

7.21.1 Changed

- Test factories and other test files are now included in the package installation. Factories can be useful when creating project tests.
- Bump allowed `lxml` to 3.6.0
- Bump allowed `python-dateutil` to 2.5.3

7.21.2 Fixes

- Don't raise on Post.tags if Post.raw_content is None

7.22 [0.3.1] - 2016-04-13

7.22.1 Added

- Support for generating `.well-known/nodeinfo` document, which was forgotten from the 0.3.0 release. Method `federation.hostmeta.generators.get_nodeinfo_well_known_document` does this task. It requires an `url` which should be the full base url of the host. Optionally `document_path` can be specified, but it is optional and defaults to the one in the NodeInfo spec.

7.23 [0.3.0] - 2016-04-13

7.23.1 Added

- Support for generating NodeInfo documents using the generator `federation.hostmeta.generators.NodeInfo`. Strict validation is skipped by default, but can be enabled by passing in `raise_on_validate` to the `NodeInfo` class. By default a warning will be generated on documents that don't conform with the strict NodeInfo values. This can be disabled by passing in `skip_validate` to the class.

7.24 [0.2.0] - 2016-04-09

7.24.1 Backwards incompatible changes

- Any implementations using the Diaspora protocol and `Post` entities must now use `DiasporaPost` instead. See "Changed" below.

7.24.2 Added

- Support for using `validate_field()` methods for entity fields and checking missing fields against `_required`. To use this validation, `validate()` must specifically be called for the entity instance.
- Base entities `Comment` and `Reaction` which subclass the new `ParticipationMixin`.
- Diaspora entity `DiasporaComment`, a variant of `Comment`.
- Diaspora entity `DiasporaLike`, a variant of `Reaction` with the `reaction = "like"` default.

7.24.3 Changed

- Refactored Diaspora XML generators into the Diaspora entities themselves. This introduces Diaspora versions of the base entities called `DiasporaPost`, `DiasporaComment` and `DiasporaLike`. **Any implementations using the Diaspora protocol and `Post` entities must now use `DiasporaPost` instead.**

7.24.4 Fixes

- Entities which don't specifically get passed a `created_at` now get correct current time in `created_at` instead of always having the time part as `00:00`.

7.25 [0.1.1] - 2016-04-03

7.25.1 Initial package release

Supports well Post type object receiving over Diaspora protocol.

Untested support for crafting outgoing protocol messages.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Index

S