# Federation Documentation

***Release 0.25.1***

**Jason Robinson**

**Mar 01, 2024**

# CONTENTS

Python library to abstract social web federation protocols like ActivityPub and Diaspora.

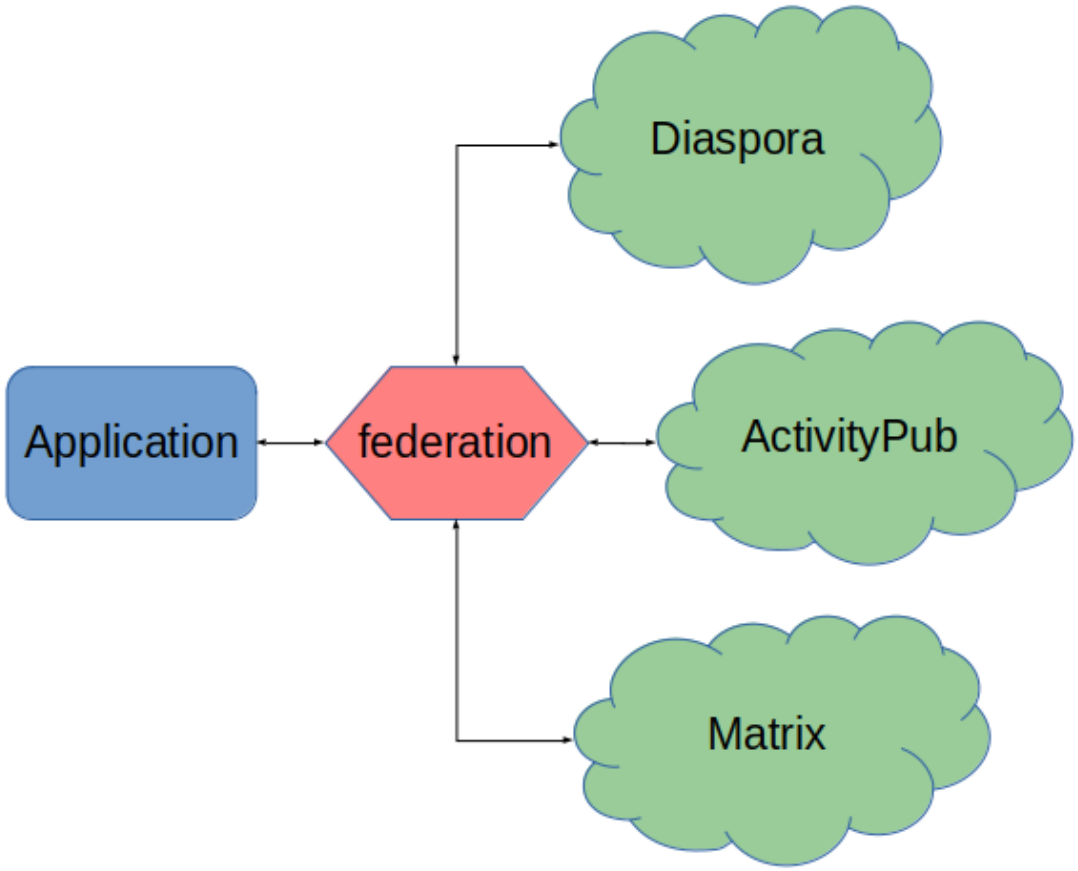
Contents:

CHAPTER

ONE

# INTRODUCTION

The aim of `federation` is to provide and abstract multiple social web protocols like ActivityPub and Diaspora in one package, over an easy to use and understand Python API. This way applications can be built to (almost) transparently support many protocols without the app builder having to know everything about those protocols.



## 1.1 Status

Currently three protocols are being focused on.

- Diaspora is considered to be stable with most of the protocol implemented.
- ActivityPub support should be considered as beta - inbound payload are handled by a jsonld processor (calamus)
- Matrix support cannot be considered usable as of yet.

The code base is well tested and in use in several projects. Backward incompatible changes 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 / @jaywink:federator.dev / GitLab / GitHub

CHAPTER

TWO

# 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

THREE

# PROTOCOLS

Currently three protocols are being focused on.

- Diaspora is considered to be stable with most of the protocol implemented.
- ActivityPub support should be considered as beta - all the basic things work and we are fixing incompatibilities as they are identified.
- Matrix support cannot be considered usable as of yet.

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

## 3.1 Diaspora

This library only supports the current renewed version of the protocol. Compatibility for the legacy version was dropped in version 0.18.0.

The feature set supported is 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

## 3.2 ActivityPub

Features currently supported:

- Webfinger
- Objects and activities as follows:
  - Actor (Person outbound, Person, Organization, Service inbound)
  - Note, Article and Page (Create, Delete, Update) * These become a `Post` or `Comment` depending on `inReplyTo`.
  - Attachment images, (inbound only for audios and videos) from the above objects
  - Follow, Accept Follow, Undo Follow
  - Announce
  - Inbound Peertube Video objects translated as `Post`.
- Inbound processing of reply collections, for platforms that implement it.
- Link, Like, View, Signature, PropertyValue, IdentityProof and Emojis objects are only processed for inbound payloads currently. Outbound processing requires support by the client application.

### 3.2.1 Namespace

All payloads over ActivityPub sent can be identified with by checking `@context` which will include the `pyfed: https://docs.jasonrobinson.me/ns/python-federation` namespace.

### 3.2.2 Content media type

The following keys will be set on the entity based on the `source` property existing:

- if the object has an `object.source` property: * `_media_type` will be the source media type (only text/markdown is supported). * `rendered_content` will be the object `content` * `raw_content` will be the source `content`
- if the object has no `object.source` property: * `_media_type` will be `text/html` * `rendered_content` will be the object `content` * `raw_content` will be empty

The `contentMap` property is processed but content language selection is not implemented yet.

For outbound entities, `raw_content` is expected to be in `text/markdown`, specifically CommonMark. The client applications are expected to provide the rendered content for protocols that require it (e.g. ActivityPub). When sending payloads, `object.contentMap` will be set to `rendered_content` and `raw_content` will be added to the `object.source` property.

### 3.2.3 Medias

Any images referenced in the `raw_content` of outbound entities will be extracted into `object.attachment` object. For receivers that don't support inline images, image attachments will have a `pyfed:inlineImage` property set to `true` to indicate the image has been extracted from the content. Receivers should ignore the inline image attachments if they support showing <img> HTML tags or the markdown content in `object.source`. Outbound audio and video attachments currently lack support from client applications.

For inbound entities we do this automatically by not including received image attachments in the entity `_children` attribute. Audio and video are passed through the client application.

### 3.2.4 Hashtags and mentions

For outbound payloads, client applications must add/set the hashtag/mention value to the `class` attribute of rendered content linkified hashtags/mentions. These will be used to help build the corresponding `Hashtag` and `Mention` objects.

For inbound payloads, if a markdown source is provided, hashtags/mentions will be extracted through the same method used for Diaspora. If only HTML content is provided, the `a` tags will be marked with a `data-[hashtag|mention]` attribute (based on the provided Hashtag/Mention objects) to facilitate the `href` attribute modifications lient applications might wish to make. This should ensure links can be replaced regardless of how the HTML is structured.

## 3.3 Matrix

The aim of Matrix support in this library is not to provide instant messaging but to wrap the parts of the Matrix protocol that specifically are especially useful for social media applications. The current ongoing work on Ceruelan provides much of what will be implemented in this library.

This library doesn't aim to be a homeserver or provide any part of the server to server API. The plan is to provide an appservice to hook onto a separate homeserver that deals with all the complex protocol related details. This library will then aim to abstract much of what the appservice gives or takes behind the same API as is provided for the other protocols.

Currently support is being added, please visit back in future versions.

NOTE! Current features also assume Django is configured, though this is likely to not be the case in the future.

### 3.3.1 Appservice

To generate the appservice registration file you must ensure you've added the relevant configuration (see *Configuration*).

Then launch a Django shell inside your project and run the following:

```
from federation.protocols.matrix.appservice import print_registration_yaml
print_registration_yaml()
```

This YAML needs to be registered with the linked Matrix homeserver as instructed in the relevant homeserver documentation.

CHAPTER

FOUR

# 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.

### 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.

If you need the correct protocol specific entity class from the base entity, each protocol will define a `get_outbound_entity` function.

### 4.1.2 Federation identifiers

All entities have an `id` to guarantee them a unique name in the network. The format of the `id` depends on the protocol in question.

- ActivityPub: maps to the object `id` (whether wrapped in an Activity or not)
- Diaspora: maps to `guid` for the entity.

#### Profiles

Profiles are uniquely identified by the `id` as above. Additionally for Diaspora they always have a `handle`. ActivityPub profiles can also have a `handle` but it is optional.

A handle will always be in email like format, without the @ prefix found on some platforms. This will be added to outgoing payloads where needed.

### 4.1.3 Creator and owner identifiers

All entities except `Profile` have an `actor_id` which tells who created this object or activity. The format depends on the protocol in question.

- ActivityPub: maps to Object `attributedTo` or Activity `actor_id`.
- Diaspora: maps to entity `author`

### 4.1.4 Activity identifiers

Entities which are an activity on something, for example creating, updating, deleting, following, etc, should have an `activity_id` given to be able to send out to the ActivityPub protocol.

### 4.1.5 Mentions

Entities store mentions in the list `_mentions`. The list is a simple list of strings which will be either an URL format `profile.id` or handle, as per above examples.

The syntax for a mention in text is URL format `@{<profile.id>}` or `@{<profile.handle>}`. The GUID format `profile.id` cannot be used for a mention.

Examples:

```
# profile.handle
Hello @{user@domain.tld}!

# profile.id in URL format
Hello @{https://domain.tld/user}
```

It is suggested `profile.handle` syntax is used always for textual mentions unless handles are not available.

#### Inbound

Mentions are added to the entity `_mentions` list when processing inbound entities. For ActivityPub they will be extracted from `Mention` tags and for Diaspora extracted from the text using the Diaspora mention format.

#### Outbound

Mentions can be given in the `_mentions` list. If not given, they will be extracted from the textual content using the above formats in the example.

For ActivityPub they will be added as `Mention` tags before sending. If the mention is in handle format, a WebFinger fetch will be made to find the profile URL format ID.

For Diaspora they will be added to the text in the correct format, if not found. If they are found in the text in non-Diaspora format, they will be converted before sending.

## 4.2 Discovery

Federation provides many generators to allow providing discovery documents. They have been made as Pythonic as possible so that library users don't have to meddle with the various documents and their internals.

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.fetchers.`**`fetch_nodeinfo_document`**(*host*)

`federation.hostmeta.fetchers.`**`fetch_nodeinfo2_document`**(*host*)

`federation.hostmeta.fetchers.`**`fetch_statisticsjson_document`**(*host*)

`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.**generate_nodeinfo2_document**(***kwargs*)

Generate a NodeInfo2 document.

Pass in a dictionary as per NodeInfo2 1.0 schema: https://github.com/jaywink/nodeinfo2/blob/master/schemas/1.0/schema.json

**Minimum required schema:**

**{server:**
baseUrl name software version

} openRegistrations

Protocols default will match what this library supports, ie "diaspora" currently.

**Returns**
dict

**Raises**
KeyError on missing required items

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.**MatrixClientWellKnown**(*homeserver_base_url: str*, *identity_server_base_url: str | None = None*, *other_keys: Dict | None = None*)

Matrix Client well-known as per https://matrix.org/docs/spec/client_server/r0.6.1#server-discovery

class federation.hostmeta.generators.**MatrixServerWellKnown**(*homeserver_domain_with_port: str*)

Matrix Server well-known as per https://matrix.org/docs/spec/server_server/r0.1.4#server-discovery

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.**RFC7033Webfinger**(*id: str*, *handle: str*, *guid: str*, *base_url: str*, *profile_path: str*, *hcard_path: str = '/hcard/users/'*, *atom_path: str | None = None*, *search_path: str | None = None*)

RFC 7033 webfinger - see https://tools.ietf.org/html/rfc7033

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

**Parameters**

- **id** – Profile ActivityPub ID in URL format
- **handle** – Profile Diaspora handle
- **guid** – Profile Diaspora guid
- **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.

## 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**(*request: RequestType*, *user: UserType | None = None*, *sender_key_fetcher: Callable[[str], str] | None = None*, *skip_author_verification: bool = False*) → Tuple[str, str, List]

Takes a request and passes it to the correct protocol.

**Returns a tuple of:**

- sender id
- 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 *request*. If this object is being relayed by the sender, the author could actually be a different identity.

**Parameters**

- **request** – Request object of type RequestType - note not a HTTP request even though the structure is similar
- **user** – User that will be passed to *protocol.receive* (only required on private encrypted content) MUST have a *private_key* and *id* 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 id, protocol name and list of entity objects

## 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_send**(*entity: BaseEntity*, *author_user: UserType*, *recipients: List[Dict]*, *parent_user: UserType | None = None*, *payload_logger: callable | None = None*) → None

Send an entity to remote servers.

Using this we will build a list of payloads per protocol. After that, each recipient will get the generated protocol payload delivered. Delivery to the same endpoint will only be done once so it's ok to include the same endpoint as a receiver multiple times.

Any given user arguments must have `private_key` and `fid` 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 dict containing at minimum the "endpoint", "fid", "public" and "protocol" keys.

  For ActivityPub and Diaspora payloads, "endpoint" should be an URL of the endpoint to deliver to.

  The "fid" can be empty for Diaspora payloads. For ActivityPub it should be the recipient federation ID should the delivery be non-private.

  The "protocol" should be a protocol name that is known for this recipient.

  The "public" value should be a boolean to indicate whether the payload should be flagged as a public payload.

  TODO: support guessing the protocol over networks? Would need caching of results

  For private deliveries to Diaspora protocol recipients, "public_key" is also required.

  For example [

  {
  "endpoint": "https://domain.tld/receive/users/1234-5678-0123-4567", "fid": "", "protocol": "diaspora", "public": False, "public_key": <RSAPublicKey object> | str,

  }, {

  "endpoint": "https://domain2.tld/receive/public", "fid": "", "protocol": "diaspora", "public": True,

  }, {

  "endpoint": "https://domain4.tld/sharedinbox/", "fid": "https://domain4.tld/profiles/jack/", "protocol": "activitypub", "public": True,

  }, {

  "endpoint": "https://domain4.tld/profiles/jill/inbox", "fid": "https://domain4.tld/profiles/jill", "protocol": "activitypub", "public": False,

  }, {

”endpoint”: “https://matrix.domain.tld”, “fid”: “#@user:domain.tld”, “protocol”: “matrix”, “public”: True,

}

]

- **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. For Activitypub, the parent_user's private key will be used to generate the http signature if the author_user is not a local user.
- **payload_logger** – (Optional) Function to log the payloads with.

## 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.

`federation.hostmeta.django.generators.`**rfc7033_webfinger_view**(*request*, **args*, ***kwargs*)

Django view to generate an RFC7033 webfinger.

`federation.hostmeta.django.generators.`**matrix_client_wellknown_view**(*request*, **args*, ***kwargs*)

`federation.hostmeta.django.generators.`**matrix_server_wellknown_view**(*request*, **args*, ***kwargs*)

`federation.hostmeta.django.generators.`**nodeinfo2_view**(*request*, **args*, ***kwargs*)

### 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,
    "federation_id": "https://example.com/u/john/",
    "get_object_function": "myproject.utils.get_object",
    "get_private_key_function": "myproject.utils.get_private_key",
    "get_profile_function": "myproject.utils.get_profile",
    "matrix_config_function": "myproject.utils.matrix_config_funct",
    "nodeinfo2_function": "myproject.utils.get_nodeinfo2_data",
    "process_payload_function": "myproject.utils.process_payload",
    "search_path": "/search/?q=",
    "tags_path": "/tags/:tag:",
}
```

- `base_url` is the base URL of the server, ie protocol://domain.tld.
- `federation_id` is a valid ActivityPub local profile id whose private key will be used to create the HTTP signature for GET requests to ActivityPub platforms.

- `get_object_function` should be the full path to a function that will return the object matching the ActivityPub ID for the request object passed to this function.
- `get_private_key_function` should be the full path to a function that will accept a federation ID (url, handle or guid) and return the private key of the user (as an RSA object). Required for example to sign outbound messages in some cases.
- `get_profile_function` should be the full path to a function that should return a `Profile` entity. The function should take one or more keyword arguments: `fid`, `handle`, `guid` or `request`. It should look up a profile with one or more of the provided parameters.
- `matrix_config_function` (optional) function that returns a Matrix configuration dictionary, with the following objects:

```
{
  # Location of the homeserver (not server name)
  "homeserver_base_url": "https://matrix.domain.tld",
  # Homeserver domain and port (not server domain)
  "homeserver_domain_with_port": "matrix.domain.tld:443",
  # Homeserver name
  "homeserver_name": "domain.tld",
  # Appservice details
  "appservice": {
    # Unique ID to register with at the homeserver. Don't change this after creating.
    "id": "uniqueid",
    # Short code (a-z only), used for various things like namespacing
    "shortcode": "federatedapp",
    # Secret token for communication
    "token": "secret_token",
  },
  # (Optional) location of identity server
  "identity_server_base_url": "https://id.domain.tld",
  # (Optional) other keys to include in the client well-known (must be a dictionary)
  "client_wellknown_other_keys": {
    "org.foo.key" "barfoo",
  },
  # (Optional) registration shared secret
  "registration_shared_secret": "supersecretstring",
}
```

- `nodeinfo2_function` (optional) function that returns data for generating a NodeInfo2 document. Once configured the path `/.well-known/x-nodeinfo2` will automatically generate a NodeInfo2 document. The function should return a `dict` corresponding to the NodeInfo2 schema, with the following minimum items:

```
{server:
    baseUrl
    name
    software
    version
}
openRegistrations
```

- `process_payload_function` (optional) function that takes in a request object. It should return `True` if successful (or placed in queue for processing later) or `False` in case of any errors.
- `search_path` (optional) site search path which ends in a parameter for search input, for example "/search?q="

- `tags_path` (optional) path format to view items for a particular tag. `:tag:` will be replaced with the tag (without #).

## 4.7 Protocols

The code for opening and creating protocol messages lives under each protocol module in `federation.protocols`.

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 ActivityPub

`federation.utils.activitypub.`**`retrieve_and_parse_content`**`(`*`**kwargs`*`)` → Any | None

`federation.utils.activitypub.`**`retrieve_and_parse_document`**`(`*`fid: str`*, *`cache: bool = True`*`)` → Any | None

Retrieve remote document by ID and return the entity.

`federation.utils.activitypub.`**`retrieve_and_parse_profile`**`(`*`fid: str`*`)` → Any | None

Retrieve the remote fid and return a Profile object.

### 4.8.2 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.`**`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: str`*, *`guid: str`*`)` → str

Get remote endpoint for delivering private payloads.

`federation.utils.diaspora.`**`get_public_endpoint`**`(`*`id: str`*`)` → str

Get remote endpoint for delivering public payloads.

federation.utils.diaspora.**parse_profile_from_hcard**(*hcard: str*, *handle: str*)

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.diaspora.entities.DiasporaProfile instance

federation.utils.diaspora.**retrieve_and_parse_content**(*id: str*, *guid: str*, *handle: str*, *entity_type: str*, *cache: bool = True*, *sender_key_fetcher: Callable[[str], str] | None = 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**

**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_diaspora_webfinger**(*handle*)

Retrieve a and parse a remote Diaspora webfinger document.

**Parameters**

**handle** – Remote handle to retrieve

**Returns**

dict

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

### 4.8.3 Matrix

federation.utils.matrix.**register_dendrite_user**(*username: str*) → Dict

Shared secret registration for Dendrite.

Note uses the legacy route, see https://github.com/matrix-org/dendrite/issues/1669

Currently compatible with Django apps only.

**Returns:**

{
'user_id': '@username:domain.tld', 'access_token': 'randomaccesstoken', 'home_server': 'domain.tld', 'device_id': 'randomdevice'
}

### 4.8.4 Network

federation.utils.network.**fetch_document**(*url=None*, *host=None*, *path='/'*, *timeout=10*, *raise_ssl_errors=True*, *extra_headers=None*, *cache=True*, ***kwargs*)

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)
- **extra_headers** – Optional extra headers dictionary to add to requests

:arg kwargs holds extra args passed to requests.get :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*, *method='post'*, **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)
- **method** – Method to use, defaults to post

**Returns**

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

### 4.8.5 Protocols

federation.utils.protocols.**identify_recipient_protocol**(*id: str*) → str | 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

FIVE

# DEVELOPMENT

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

- Source code repo
- Issue tracker

## 5.1 Environment setup

Once you have your (Python 3.7+) 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 Matrix on room `#socialhome:federator.dev`.

You can also ask questions or give feedback via issues.

CHAPTER

SIX

# 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.
- The Federation info - statistics and node list for the federated web.

# Changelog

## [0.25.1] - 2024-02-18

### Fixed

- Address CVE-2024-23832 by ensuring that a pulled AP payload id netloc is the same as the request fid netloc.

## [0.25.0] - 2024-01-06

### Added

- LD signature. Relayable AP payloads signatures are checked (inbound) and signed (outbound). A missing or invalid signature on inbound payloads will trigger a fetch if the sender differs from the author (i.e., a relay).
- The *signable* attribute has been added. It defaults to *False* and will enforce the fetching of relayed payloads with a bad signature when set to `True`on a given class.
- The *url* property is now set to the *id* property as some platforms make use of it.

### Changed

- Re-implement dynamically generated LD contexts for outbound payloads. AP extensions are defined on a per class/property basis. For classes, a *ctx* attribute is set if required. For properties, the calamus field *metadata* property is used.
- For inbound payload, a cached dict of all the defined AP extensions is merged with each incoming LD context.
- Better handle conflicting property defaults by having *get_base_attributes* return only attributes that are not empty (or bool). This helps distinguish between *marshmallow.missing* and empty values.
- JsonLD document caching now set in *activitypub/__init__.py*.
- Patch outbound payloads for platform that don't handle arrays compacted to a single value and *as:Public*.
- Always try to get profiles from the client app before fetching from remote. In support of this, the client app AP profiles must include the keyId and the followers URIs. As a significant side effect, profile retractions are now more likely to succeed.
- Switch to BeautifulSoup for content parsing. The client app is now expected to provide the rendered content for outbound payloads. Mark inbound AP payload hashtag and mention links and let the client app deal with them.

- Move process_text_links back to the client app.
- Handle gotosocial reply collections.

### Fixed

- Inbound AP share retractions (undo announce) were deserialized as a *base.Retraction* class, which would throw an error when accessing the missing *signable* attribute. To fix this, a *Retraction* class was added.
- Because of the additions and changes above, a number of tests needed to be fixed.
- HTTP signature verification now returns the signature author fid which is used as the actual sender by *message_to_object*.
- In fetch_document: if response.encoding is not set, default to utf-8.
- Fix process_text_links that would crash on *a* tags with no *href* attribute.
- Ignore relayed AP retractions.
- Fix AP profile processing for hubzilla, guppe and bird.makeup.
- Unquote and normalize hashtag links.
- Fix Peertube payload processing when the content property is missing.
- Ensure the outbound AP profile to property is an array.

## [0.24.1] - 2023-03-18

### Fixed

- Fix documentation builds

## [0.24.0] - 2023-03-18

### Added

- Add a validation function for the Activitypub *attributedTo* property. Ensure it starts with *http*.

### Changed

- Optimize handle_send by ensuring a payload is only sent once per recipient unique endpoint.
- Match the Activitypub Hashtag object *href* property value against the raw content in order to make this process platform agnostic.

### Fixed

- The Activitypub *url* property can now handle nested Link objects for all defined object types.
- Catch cases where an Activitypub CollectionPage *next* property points back to a Collection object.
- Make the Activitypub Follow class handle both the Undo and the Accept activities.

## [0.23.1] - 2023-02-08

### Changed

- Switch *python-httpsig-socialhome* dependency to PyPi packaged version.

## [0.23.0] - 2023-02-08

### Added

- Activitypub payloads are now processed by calamus (https://github.com/SwissDataScienceCenter/calamus), which is a jsonld processor based on marshmallow.
  - A large number of inbound Activitypub objects and properties are deserialized, it's up to the client app to implement the corresponding behavior.

  - Unsupported objects and properties should be easy to implement. Unsupported payloads are logged as such.
  - More AP platforms are now supported (friendica, pixelfed, misskey, pleroma, gotosocial, litepub, and more). The jsonld context some platforms provide sometimes needs to be patched because of missing jsonld term definitions.
  - Peertube Video objects are translated into Posts.
- For performance, requests_cache has been added. It pulls a redis configuration from django if one exists or falls back to a sqlite backend. Special case: pyld document loader has been extended to use redis directly.
- Activitypub GET requests are now signed if the django configuration includes FEDERATION_USER which is used to fetch that user's private key.
- Activitypub remote GET signature is now verified in order to authorize remote access to limited content.
- Added Video and Audio objects. Inbound support only.
- Process Activitypub reply collections. When supported by the client app, it allows for a more complete view of conversations, especially for shared content.
- WIP: initial support for providing reponses to Activitypub collections requests. This release only responds with a count for the followers and following collections.

### Changed

- outbound.py doesn't need to set the to and cc Activitypub properties, they are now expected to be set by the client app.
- Attempts are made to remove duplicate img tags some platforms send (friendica, for one).
- Activitypub receivers of the followers variant are now correctly processed for all known platforms.
- Accept images with application/octet-stream content type (with the help of the magic library).
- user@domain is now the only format used for mentions. The client app is expected to comply. For Activitypub, this means making a webfinger request to validate the handle if the client app doesn't already know the corresponding profile.
- Because of the change above, ensure mentions in Diaspora outbound payloads are as per their protocol spec (i.e. replacing @user@domain with @{user@domain} in the text)

### Fixed

- Signatures are not verified and the corresponding payload is dropped if no public key is found.
- Sign forwarded AP replies and shares with the target content author's private key.

### Internal changes

- Dropped python 3.6 support.
- Many tests were fixed/updated.

## [0.22.0] - 2021-08-15

### Added

- Work in progress Matrix support over an appservice

  Currently requires Django support. Tested on Dendrite and up to version v0.3.11 only. Features so far:

  - Register local users on the configured Matrix server.
  - Post local user public posts into Matrix side to their profile timeline rooms and to each hashtag room.

### Fixed

- Fixed image delivery between platforms that send ActivityPub payloads with a markdown *source*, caused by overenthusiastic linkifying of markdown.
- Fix a crash in *outbound.handle_send* when payload failed to be generated and *parent_user* was not given.

## [0.21.0] - 2020-12-20

### Added

- Start testing on Python 3.8 which is the new recommended version to use.

### Removed

- Removed the network utils *fetch_host_ip_and_country* and *fetch_country_by_ip* due to the library that was used starting to require an API key.

### Internal changes

- Fix some tests for newer Python.

## [0.20.0] - 2020-12-20

### Added

- Entities with a *raw_content* field now have URL syntax mentions rendered into a link. ([related issue](https://git.feneas.org/socialhome/socialhome/issues/572))

  If Django is configured, a profile will be retrieved using the configured profile getter function and the profile name or username will be used for the link.
- Add *process_text_links* text utility to linkify URL's in text.
- Add *find_tags* text utility to find hashtags from text. Optionally the function can also replace the tags through a given *replacer* function. This utility is used to improve the tag extraction logic from entities text fields. ([related issue](https://git.feneas.org/jaywink/federation/issues/70))
- Outbound functions *handle_send* and *handle_create_payload* now accept an optional *payload_logger* parameter. If given it should be a function that takes three parameters:
  - *str* or *dict* payload
  - *str* protocol name
  - *str* sender id

  The function will be called for each generated payload.
- **Cross-protocol improvements:**
  - Extract Diaspora guid from ActivityPub payloads implementing the Diaspora extension.
  - Add Diaspora extension and guid to outbound ActivityPub payloads, if available. For profiles, also add handle.
  - Extract ActivityPub ID from Diaspora payloads if found as the *activitypub_id* property.
  - Add ActivityPub ID to outbound Diaspora payloads of types comment, post and profile, if an URL given as *id*.

### Changed

- The NodeInfo2 hostmeta parser now cleans the port out of the host name.
- URL's in outgoing text content are now linkified for the HTML representation of the content for ActivityPub payloads.
- Don't include OStatus for Mastodon 3.0+ protocols list. ([related issue](https://github.com/thefederationinfo/the-federation.info/issues/217))

- **Backwards incompatible**: Stop markdownifying incoming ActivityPub content. Instead copy it as is to the `raw_content` attribute on the entity, setting also the `_media_type` to `text/html`.

### Fixed

- Don't crash loudly when fetching webfinger for Diaspora that does not contain XML.
- Add missing *response.raise_for_status()* call to the *fetch_document* network helper when fetching with given URL. Error status was already being raised correctly when fetching by domain and path.
- Don't crash when parsing an invalid NodeInfo document where the usage dictionary is not following specification.
- Ensure Pixelfed, Kroeg and Kibou instances that emulate the Mastodon API don't get identified as Mastodon instances.
- Loosen validation of *TargetIDMixin*, it now requires one of the target attributes to be set, not just *target_id*. This fixes follows over the Diaspora protocol which broke with stricter send validation added in 0.19.0.
- Fix some edge case crashes of *handle_send* when there are Diaspora protocol receivers.
- Fix reading *sharedInbox* from remote ActivityPub profiles. This caused public payloads not to be deduplicated when sending public payloads to remote ActivityPub servers. Refetching profiles should now fix this. ([related issue](https://git.feneas.org/jaywink/federation/issues/124))
- Don't always crash generating payloads if Django is installed but not configured.
- Don't try to relay AP payloads to Diaspora receivers and vice versa, for now, until cross-protocol relaying is supported.
- Fix some characters stopping tags being identified ([related issue](https://git.feneas.org/socialhome/socialhome/-/issues/222))
- Fix tags separated by slashes being identified ([related issue](https://git.feneas.org/socialhome/socialhome/-/issues/198))

## [0.19.0] - 2019-12-15

### Added

- The fetcher *retrieve_remote_profile* now also supports handle based fetching for the ActivityPub protocol.

### Changed

- All outgoing entities are now validated before sending. This stops the sending of invalid entities to the network, for example a Share of a Post from ActivityPub to the Diaspora protocol network.

### Fixed

- Allow ActivityPub HTTP Signature verification to pass if signature is at most 24 hours old.

  Previously requirement was 30 seconds, which caused loss of messages where signature validation didn't happen immediately, but in a background worker which didn't immediately process the job.

### Internal changes

- Improve performance of generating ActivityPub payloads for a large number of receivers in *handle_send*.
- Fail early in outbound *handle_send* if a payload cannot be generated for a payload which doesn't depend on recipient attributes.

## [0.18.1] - 2019-10-06

### Changed

- Removed possibility to deactivate ActivityPub support. It is now always enabled by default.

## [0.18.0] - 2019-10-06

### Added

- Base entities *Post*, *Comment* and *Image* now accept an *url* parameter. This will be used when serializing the entities to AS2 for ActivityPub.
- RFC7033 webfinger generator now has compatibility to platforms using it with ActivityPub. It now lists *aliases* pointing to the ActivityPub entity ID and profile URL. Also there is a *rel=self* to point to the *application/activity+json* AS2 document location.
- Added a Django view decorator that makes any Profile or Post view ActivityPub compatible. Right now basic AS2 serialization is supported when the view is called using the supported content types in the Accept header. If the content types are not in the header, the view will render normally.

  **When used, a few extra settings must be given in the Django *FEDERATION* configuration dictionary.**

  - *get_object_function* should contain the Python path to a function that takes a request object and returns an object matching the ActivityPub ID for the request or *None*.
  - *process_payload_function* should contain the Python path to a function that takes in a request object. It should return *True* if successful (or placed in queue for processing later) or *False* in case of any errors.
- Added network utility *network.fetch_host_ip* to fetch IP by hostname.
- Entities of type *Profile* now have a dictionary of *inboxes*, with two elements, *private* and *public*. These should be URL's indicating where to send payloads for the recipient.

  ActivityPub profiles will parse these values from incoming profile documents. Diaspora entities will default to the inboxes in the specification.
- Added support for Diaspora *Comment* entity *thread_parent_guid* attribute.
- Added *root_target_id* and *root_target_guid* to *Comment* base entity. This allows referring to a parent object up the hierarchy chain for threaded comments.
- The high level fetcher *retrieve_remote_content* now supports ActivityPub ID's.
- All ActivityPub payloads are added a *pyfed: https://docs.jasonrobinson.me/ns/python-federation* context to identify payloads sent by this library.
- Entities with *raw_content* now also contain a *_media_type* and *rendered_content*.

  The default *_media_type* is *text/markdown* except for ActivityPub originating posts it defaults to *text/html*. If the ActivityPub payload contains a *source*, that mediaType will be used instead.
- Host meta fetchers now support NodeInfo 2.1

### Changed

- **Backwards incompatible.** Lowest compatible Python version is now 3.6.
- **Backwards incompatible.** Internal refactoring to allow adding ActivityPub support as the second supported protocol. Highlights of changes below.
  - Reversal of all the work previously done to use Diaspora URL format identifiers. Working with the Diaspora protocol now always requires using handles and GUID's as before the changes introduced in v0.15.0. It ended up impossible to construct a Diaspora URL in all cases in a way that apps only need to store one identifier.
  - The *id* and possible *target_id* are now either URL format identifiers (ActivityPub) or a handle or GUID (Diaspora, depending on entity). Additionally a new *actor_id* has been added which for ActivityPub is an URL and for Diaspora a handle. Note, Diaspora entities always have also the *guid*, *handle*, *target_guid* and

*target_handle* as before v0.15.0, depending on the entity. When creating Diaspora entities, you must pass these in for sending to work.

- The high level *fetchers.retrieve_remote_content* signature has changed. It now expects an *id* for fetching from AP protocol and *handle*, *guid* and *entity_type* to fetch from Diaspora. Additionally a *sender_key_fetcher* can be passed in as before to optimize public key fetching using a callable.
- The high level *fetchers.retrieve_remote_profile* signature has changed. It now expects as first parameter an *id* which for ActivityPub objects is the URL ID and for Diaspora objects is the handle. Additionally a *sender_key_fetcher* can be passed in as before to optimize public key fetching using a callable.
- The generator class *RFC7033Webfinger* now expects instead of an *id* the *handle* and *guid* of the profile.
- NodeInfo2 parser now returns the admin user in *handle* format instead of a Diaspora format URL.
- The high level inbound and outbound functions *inbound.handle_receive*, *outbound.handle_send* parameter *user* must now receive a *UserType* compatible object. This must have the attribute *id*, and for *handle_send* also *private_key*. If Diaspora support is required then also *handle* and *guid* should exist. The type can be found as a class in *types.UserType*.
- The high level inbound function *inbound.handle_receive* first parameter has been changed to *request* which must be a *RequestType* compatible object. This must have the attribute *body* which corrresponds to the old *payload* parameter. For ActivityPub inbound requests the object must also contain *headers*, *method* and *url*.
- The outbound function *outbound.handle_send* parameter *recipients* structure has changed. It must now be a list of dictionaries, containing at minimum the following: *endpoint* for the recipient endpoint, *fid* for the recipient federation ID (ActivityPub only), *protocol* for the protocol to use and *public* as a boolean whether the payload should be treated as visible to anyone.

  For Diaspora private deliveries, also a *public_key* is required containing the receiver public key. Note that passing in handles as recipients is not any more possible - always pass in a url for *endpoint*.
- The outbound function *outbound.handle_create_payload* now requires an extra third parameter for the protocol to use. This function should rarely need to be called directly - use *handle_send* instead which can handle both ActivityPub and Diaspora protocols.
- The *Image* base entity has been made more generic.

  The following were removed: *remote_path*, *remote_name*, *linked_type*, *linked_guid*, *public*.

  The following were added: *url*, *name*.

- **Backwards incompatible.** Generator *RFC3033Webfinger* and the related *rfc3033_webfinger_view* have been renamed to *RFC7033Webfinger* and *rfc7033_webfinger_view* to reflect the right RFC number.
- Network helper utility *fetch_document* can now also take a dictionary of *headers*. They will be passed to the underlying *requests* method call as is.
- *Retraction* entity can now also have an *entity_type* of *Object*. Receivers will need to find the correct object using *target_id* only. This is currently only relevant for ActivityPub where retraction messages do not refer to object type.
- **Backwards incompatible.** Inbound entities now have a list of receivers.

  Entities processed by inbound mappers will now have a list of receivers in *_receivers*. This replaces the *_receiving_actor_id* which was previously set for Diaspora entities.
- UserType now has a *receiver_variant* which is one of *ReceiverVariant* enum. *ACTOR* means this receiver is a single actor ID. *FOLLOWERS* means this is the followers of the ID in the receiver.

### Fixed

- Ensure Diaspora mentions are extracted when they don't have a display name part.

### Removed

- **Backwards incompatible.** Support for Legacy Diaspora payloads have been removed to reduce the amount of code needed to maintain while refactoring for ActivityPub.

## [0.17.0] - 2018-08-11

### Fixed

- Switch crypto library *pycrypto* to *pycryptodome*, which is a more up to date fork of the former. This fixes CVE-2018-6594 found in the former.

  **Deployment note.** When updating an application, you *must* uninstall *pycrypto* first, otherwise there will be a conflict if both the versions are installed at the same time. To uninstall, do *pip uninstall pycrypto*.

## [0.16.0] - 2018-07-23

### Added

- Enable generating encrypted JSON payloads with the Diaspora protocol which adds private message support. ([related issue](https://github.com/jaywink/federation/issues/82))

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

- Add RFC7033 webfinger generator ([related issue](https://github.com/jaywink/federation/issues/108))

  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.

- Add fetchers and parsers for NodeInfo, NodeInfo2, StatisticsJSON and Mastodon server metainfo documents.

- Add NodeInfo2 generator and Django view. See documentation for details. ([related issue](https://github.com/jaywink/federation/issues/32))

- Added new network utilities to fetch IP and country information from a host.

  The country information is fetched using the free *ipdata.co* service. NOTE! This service is rate limited to 1500 requests per day.

- Extract mentions from Diaspora payloads that have text content. The mentions will be available in the entity as *_mentions* which is a set of Diaspora ID's in URI format.

### Changed

- Send outbound Diaspora payloads in new format. Remove possibility to generate legacy MagicEnvelope payloads. ([related issue](https://github.com/jaywink/federation/issues/82))

- **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](https://github.com/jaywink/federation/issues/82))

  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.

- Switch Diaspora protocol to send new style entities ([related issue](https://github.com/jaywink/federation/issues/59))

We've already accepted these on incoming payloads for a long time and so do all the other platforms now, so now we always send out entities with the new property names. This can break federation with really old servers that don't understand these keys yet.

### Fixed

- Change unquote method used when preparing Diaspora XML payloads for verification ([related issue](https://github.com/jaywink/federation/issues/115))

  Some platforms deliver payloads not using the urlsafe base64 standard which caused problems when validating the unquoted signature. Ensure maximum compatibility by allowing non-standard urlsafe quoted payloads.

- Fix for empty values in Diaspora protocol entities sometimes ending up as *None* instead of empty string when processing incoming payloads.
- Fix validation of *Retraction* with entity type *Share*
- Allow port in Diaspora handles as per the protocol specification

  Previously handles were validated like emails.

- Fix Diaspora *Profile* mapping regarding *last_name* property

  Previously only *first_name* was used when creating the *Profile.name* value. Now both *first_name* and *last_name* are used.

  When creating outgoing payloads, the *Profile.name* will still be placed in *first_name* to avoid trying to artificially split it.

## [0.15.0] - 2018-02-12

### Added * Added base entity *Share* which maps to a *DiasporaReshare* for the Diaspora protocol. ([related issue](https://github.com/jaywink/federation/issues/94))

  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](https://diaspora.github.io/diaspora_federation/federation/diaspora_scheme.html), 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](https://github.com/jaywink/federation/issues/103))

  This function takes the following parameters:

  - *id* - Object ID. For Diaspora, the only supported protocol at the moment, this is in the [Diaspora URI](https://diaspora.github.io/diaspora_federation/federation/diaspora_scheme.html) 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](https://github.com/jaywink/federation/issues/108))

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

### 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.

### 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.

## [0.14.1] - 2017-08-06

### 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.

## [0.14.0] - 2017-08-06

### 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.

### 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.

## [0.13.0] - 2017-07-22

### 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()*.

### 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](https://github.com/jaywink/federation/issues/83))

- 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](https://github.com/jaywink/federation/issues/59))

### 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](https://github.com/jaywink/federation/issues/89)) * 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](https://github.com/jaywink/federation/issues/88)) * Fix processing Diaspora payload if it is passed to *handle_receive* as a *bytes* object. ([issue](https://github.com/jaywink/federation/issues/91)) * 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.

### 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.

## [0.12.0] - 2017-05-22

### 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.

### Added * New style Diaspora public payloads are now supported (see [here](https://github.com/diaspora/diaspora_federation/issues/30)). 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.

### 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.

## [0.11.0] - 2017-05-08

### 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*.

### 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.

### 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.

## [0.10.1] - 2017-03-09

### Fixes * Ensure tags are lower cased after collecting them from entity *raw_content*.

## [0.10.0] - 2017-01-28

### 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.

### Fixes * Don't crash *federation.utils.diaspora.retrieve_diaspora_webfinger* if XRD parse raises an *xml.parsers.expat.ExpatError*.

## [0.9.1] - 2016-12-10

### Fixes * Made *Profile.raw_content* optional. This fixes validating profiles parsed from Diaspora hCard's.

## [0.9.0] - 2016-12-10

### 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.

### 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.

### 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.

## [0.8.2] - 2016-10-23

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

## [0.8.1] - 2016-10-18

### 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*.

## [0.8.0] - 2016-10-09

### 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.

### 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.

### Added * Added *Retraction* entity with *DiasporaRetraction* counterpart.

## [0.7.0] - 2016-09-15

### 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.

### 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.

### 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.

## [0.6.1] - 2016-09-14

### 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.

## [0.6.0] - 2016-09-13

### 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.

### 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.

### 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.

## [0.5.0] - 2016-09-05

### 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](https://github.com/jaywink/federation/pull/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`**.

### Added - *Post.provider_display_name* is now supported in the entity outbound/inbound mappers. [#44](https://github.com/jaywink/federation/pull/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](https://github.com/jaywink/federation/pull/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](https://github.com/jaywink/federation/pull/46)

## [0.4.1] - 2016-09-04

### 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](https://github.com/jaywink/federation/pull/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](https://github.com/jaywink/federation/pull/41)

## [0.4.0] - 2016-07-24

### 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.

### 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.

### 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.

### 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.

## [0.3.2] - 2016-05-09

### 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

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

## [0.3.1] - 2016-04-13

### 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.

## [0.3.0] - 2016-04-13

### Added - Support for generating [NodeInfo](http://nodeinfo.diaspora.software) 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.

## [0.2.0] - 2016-04-09

### Backwards incompatible changes - Any implementations using the Diaspora protocol and *Post* entities must now use *DiasporaPost* instead. See "Changed" below.

### 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.

### 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.**

### 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*.

## [0.1.1] - 2016-04-03

### Initial package release

Supports well Post type object receiving over Diaspora protocol.

Untested support for crafting outgoing protocol messages.

CHAPTER

SEVEN

# INDICES AND TABLES

- genindex
- modindex
- search

# INDEX

## S