acme-python Documentation

Release 0

Let's Encrypt Project

Mar 31, 2020

Contents

Contents:

CHAPTER 1

API Documentation

1.1 Challenges

ACME Identifier Validation Challenges.

class acme.challenges.**Challenge**(***kwargs*)
Bases: josepy.json_util.TypedJSONObjectWithFields

ACME challenge.

classmethod from_json(*jobj*)
Deserialize ACME object from valid JSON object.

Raises **josepy.errors.UnrecognizedTypeError** – if type of the ACME object has not been registered.

class acme.challenges.**ChallengeResponse**(***kwargs*)
Bases: acme.mixins.ResourceMixin, acme.mixins.TypeMixin, josepy.json_util.TypedJSONObjectWithFields

ACME challenge response.

class acme.challenges.**UnrecognizedChallenge**(*jobj*)
Bases: *acme.challenges.Challenge*

Unrecognized challenge.

ACME specification defines a generic framework for challenges and defines some standard challenges that are implemented in this module. However, other implementations (including peers) might define additional challenge types, which should be ignored if unrecognized.

Variables **jobj** – Original JSON decoded object.

to_partial_json()
Get JSON serializable object.

Returns Serializable JSON object representing ACME typed object. `validate()` will almost certainly not work, due to reasons explained in `josepy.interfaces.IJSONSerializable`.

Return type dict

classmethod from_json(*jobj*)
Deserialize ACME object from valid JSON object.

Raises **josepy.errors.UnrecognizedTypeError** – if type of the ACME object has not been registered.

class acme.challenges.**KeyAuthorizationChallengeResponse**(***kwargs*)
Bases: *acme.challenges.ChallengeResponse*

Response to Challenges based on Key Authorization.

Parameters **key_authorization** (*unicode*) –

verify(*chall*, *account_public_key*)
Verify the key authorization.

Parameters

- **chall** (*KeyAuthorization*) – Challenge that corresponds to this response.
- **account_public_key** (*JWK*) –

Returns `True` iff verification of the key authorization was successful.

Return type bool

to_partial_json()
See josepy.JSONDeserializable.to_partial_json()

class acme.challenges.**KeyAuthorizationChallenge**(***kwargs*)
Bases: acme.challenges._TokenChallenge

Challenge based on Key Authorization.

Parameters

- **response_cls** – Subclass of *KeyAuthorizationChallengeResponse* that will be used to generate *response*.
- **typ** (*str*) – type of the challenge

key_authorization(*account_key*)
Generate Key Authorization.

Parameters **account_key** (*JWK*) –

Rtype unicode

response(*account_key*)
Generate response to the challenge.

Parameters **account_key** (*JWK*) –

Returns Response (initialized `response_cls`) to the challenge.

Return type *KeyAuthorizationChallengeResponse*

validation(*account_key*, ***kwargs*)
Generate validation for the challenge.

Subclasses must implement this method, but they are likely to return completely different data structures, depending on what's necessary to complete the challenge. Interpretation of that return value must be known to the caller.

Parameters **account_key** (*JWK*) –

Returns Challenge-specific validation.

response_and_validation(*account_key*, **args*, ***kwargs*)
Generate response and validation.

Convenience function that return results of *response* and *validation*.

Parameters **account_key** (*JWK*) –

Return type tuple

class acme.challenges.**DNS01Response**(***kwargs*)
Bases: *acme.challenges.KeyAuthorizationChallengeResponse*

ACME dns-01 challenge response.

simple_verify(*chall*, *domain*, *account_public_key*)
Simple verify.

This method no longer checks DNS records and is a simple wrapper around *KeyAuthorizationChallengeResponse.verify*.

Parameters

- **chall** (challenges.DNS01) – Corresponding challenge.
- **domain** (*unicode*) – Domain name being verified.
- **account_public_key** (*JWK*) – Public key for the key pair being authorized.

Returns True iff verification of the key authorization was successful.

Return type bool

class acme.challenges.**DNS01**(***kwargs*)
Bases: *acme.challenges.KeyAuthorizationChallenge*

ACME dns-01 challenge.

response_cls
alias of *DNS01Response*

LABEL = '_acme-challenge'
Label clients prepend to the domain name being validated.

validation(*account_key*, ***unused_kwargs*)
Generate validation.

Parameters **account_key** (*JWK*) –

Return type unicode

validation_domain_name(*name*)
Domain name for TXT validation record.

Parameters **name** (*unicode*) – Domain name being validated.

class acme.challenges.**HTTP01Response**(***kwargs*)
Bases: *acme.challenges.KeyAuthorizationChallengeResponse*

ACME http-01 challenge response.

PORT = 80
Verification port as defined by the protocol.

You can override it (e.g. for testing) by passing `port` to *simple_verify*.

WHITESPACE_CUTSET = '\n\r\t '
Whitespace characters which should be ignored at the end of the body.

simple_verify(*chall*, *domain*, *account_public_key*, *port=None*)
Simple verify.

Parameters

- **chall** (*challenges.SimpleHTTP*) – Corresponding challenge.
- **domain** (*unicode*) – Domain name being verified.
- **account_public_key** (*JWK*) – Public key for the key pair being authorized.
- **port** (*int*) – Port used in the validation.

Returns `True` iff validation with the files currently served by the HTTP server is successful.

Return type bool

class acme.challenges.**HTTP01**(**kwargs*)
Bases: *acme.challenges.KeyAuthorizationChallenge*

ACME http-01 challenge.

response_cls
alias of *HTTP01Response*

URI_ROOT_PATH = '.well-known/acme-challenge'
URI root path for the server provisioned resource.

path
Path (starting with '/') for provisioned resource.

Return type string

uri(*domain*)
Create an URI to the provisioned resource.

Forms an URI to the HTTPS server provisioned resource (containing `token`).

Parameters **domain** (*unicode*) – Domain name being verified.

Return type string

validation(*account_key*, ***unused_kwargs*)
Generate validation.

Parameters **account_key** (*JWK*) –

Return type unicode

class acme.challenges.**TLSALPN01Response**(**kwargs*)
Bases: *acme.challenges.KeyAuthorizationChallengeResponse*

ACME tls-alpn-01 challenge response.

PORT = 443
Verification port as defined by the protocol.

You can override it (e.g. for testing) by passing `port` to *simple_verify*.

h
Hash value stored in challenge certificate

gen_cert(*domain*, *key=None*, *bits=2048*)
Generate tls-alpn-01 certificate.

Parameters

- **domain** (*unicode*) – Domain verified by the challenge.
- **key** (*OpenSSL.crypto.PKey*) – Optional private key used in certificate generation. If not provided (`None`), then fresh key will be generated.
- **bits** (*int*) – Number of bits for newly generated key.

Return type `tuple` of `OpenSSL.crypto.X509` and `OpenSSL.crypto.PKey`

probe_cert(*domain*, *host=None*, *port=None*)
Probe tls-alpn-01 challenge certificate.

Parameters

- **domain** (*unicode*) – domain being validated, required.
- **host** (*string*) – IP address used to probe the certificate.
- **port** (*int*) – Port used to probe the certificate.

verify_cert(*domain*, *cert*)
Verify tls-alpn-01 challenge certificate.

Parameters

- **domain** (*unicode*) – Domain name being validated.
- **cert** (*OpensSSL.crypto.X509*) – Challenge certificate.

Returns Whether the certificate was successfully verified.

Return type bool

simple_verify(*chall*, *domain*, *account_public_key*, *cert=None*, *host=None*, *port=None*)
Simple verify.

Verify `validation` using `account_public_key`, optionally probe tls-alpn-01 certificate and check using *verify_cert*.

Parameters

- **chall** (`challenges.TLSALPN01`) – Corresponding challenge.
- **domain** (*str*) – Domain name being validated.
- **account_public_key** (*JWK*) –
- **cert** (*OpenSSL.crypto.X509*) – Optional certificate. If not provided (`None`) certificate will be retrieved using *probe_cert*.
- **host** (*string*) – IP address used to probe the certificate.
- **port** (*int*) – Port used to probe the certificate.

Returns `True` if and only if client's control of the domain has been verified.

Return type bool

class acme.challenges.**TLSALPN01**(***kwargs*)
Bases: *acme.challenges.KeyAuthorizationChallenge*

ACME tls-alpn-01 challenge.

response_cls
alias of *TLSALPN01Response*

validation(*account_key*, ***kwargs*)
Generate validation.

Parameters

- **account_key** (*JWK*) –
- **domain** (*unicode*) – Domain verified by the challenge.
- **cert_key** (*OpenSSL.crypto.PKey*) – Optional private key used in certificate generation. If not provided (None), then fresh key will be generated.

Return type tuple of OpenSSL.crypto.X509 and OpenSSL.crypto.PKey

static is_supported()
Check if TLS-ALPN-01 challenge is supported on this machine. This implies that a recent version of OpenSSL is installed (>= 1.0.2), or a recent cryptography version shipped with the OpenSSL library is installed.

Returns True if TLS-ALPN-01 is supported on this machine, False otherwise.

Return type bool

class acme.challenges.**DNS**(***kwargs*)
Bases: acme.challenges._TokenChallenge

ACME "dns" challenge.

LABEL = '_acme-challenge'
Label clients prepend to the domain name being validated.

gen_validation(*account_key*, *alg=RS256*, ***kwargs*)
Generate validation.

Parameters

- **account_key** (*JWK*) – Private account key.
- **alg** (*JWA*) –

Returns This challenge wrapped in JWS

Return type JWS

check_validation(*validation*, *account_public_key*)
Check validation.

Parameters

- **validation** (*JWS*) –
- **account_public_key** (*JWK*) –

Return type bool

gen_response(*account_key*, ***kwargs*)
Generate response.

Parameters

- **account_key** (*JWK*) – Private account key.
- **alg** (*JWA*) –

Return type *DNSResponse*

validation_domain_name(*name*)
Domain name for TXT validation record.

Parameters **name** (*unicode*) – Domain name being validated.

class acme.challenges.**DNSResponse**(***kwargs*)
Bases: *acme.challenges.ChallengeResponse*

ACME "dns" challenge response.

Parameters **validation** (*JWS*) –

check_validation(*chall*, *account_public_key*)
Check validation.

Parameters

- **chall** (challenges.DNS) –
- **account_public_key** (*JWK*) –

Return type bool

1.2 Client

ACME client API.

class acme.client.**ClientBase**(*directory*, *net*, *acme_version*)
Bases: object

ACME client base object.

Variables

- **directory** (messages.Directory) –
- **net** (ClientNetwork) – Client network.
- **acme_version** (*int*) – ACME protocol version. 1 or 2.

update_registration(*regr*, *update=None*)
Update registration.

Parameters

- **regr** (messages.RegistrationResource) – Registration Resource.
- **update** (messages.Registration) – Updated body of the resource. If not provided, body will be taken from `regr`.

Returns Updated Registration Resource.

Return type *RegistrationResource*

deactivate_registration(*regr*)
Deactivate registration.

Parameters **regr** (messages.RegistrationResource) – The Registration Resource to be deactivated.

Returns The Registration resource that was deactivated.

Return type *RegistrationResource*

deactivate_authorization(*authzr*)
Deactivate authorization.

Parameters **authzr** (messages.AuthorizationResource) – The Authorization resource to be deactivated.

Returns The Authorization resource that was deactivated.

Return type *AuthorizationResource*

answer_challenge(*challb*, *response*)
Answer challenge.

Parameters

- **challb** (*ChallengeBody*) – Challenge Resource body.
- **response** (*challenges.ChallengeResponse*) – Corresponding Challenge response

Returns Challenge Resource with updated body.

Return type *ChallengeResource*

Raises ***UnexpectedUpdate*** –

classmethod retry_after(*response*, *default*)
Compute next `poll` time based on response `Retry-After` header.

Handles integers and various datestring formats per https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.37

Parameters

- **response** (*requests.Response*) – Response from `poll`.
- **default** (*int*) – Default value (in seconds), used when `Retry-After` header is not present or invalid.

Returns Time point when next `poll` should be performed.

Return type datetime.datetime

class acme.client.**Client**(*directory*, *key*, *alg=RS256*, *verify_ssl=True*, *net=None*)
Bases: *acme.client.ClientBase*

ACME client for a v1 API.

Variables

- **directory** (messages.Directory) –
- **key** – josepy.JWK (private)
- **alg** – josepy.JWASignature
- **verify_ssl** (*bool*) – Verify SSL certificates?
- **net** (ClientNetwork) – Client network. Useful for testing. If not supplied, it will be initialized using `key`, `alg` and `verify_ssl`.

register(*new_reg=None*)
Register.

Parameters **new_reg** (NewRegistration) –

Returns Registration Resource.

Return type *RegistrationResource*

query_registration(*regr*)
Query server about registration.

Parameters **messages.RegistrationResource** – Existing Registration Resource.

agree_to_tos(*regr*)
Agree to the terms-of-service.

Agree to the terms-of-service in a Registration Resource.

Parameters **regr** (*RegistrationResource*) – Registration Resource.

Returns Updated Registration Resource.

Return type *RegistrationResource*

request_challenges(*identifier*, *new_authzr_uri=None*)
Request challenges.

Parameters

- **identifier** (messages.Identifier) – Identifier to be challenged.
- **new_authzr_uri** (*str*) – Deprecated. Do not use.

Returns Authorization Resource.

Return type *AuthorizationResource*

Raises ***errors.WildcardUnsupportedError*** – if a wildcard is requested

request_domain_challenges(*domain*, *new_authzr_uri=None*)
Request challenges for domain names.

This is simply a convenience function that wraps around *request_challenges*, but works with domain names instead of generic identifiers. See request_challenges for more documentation.

Parameters

- **domain** (*str*) – Domain name to be challenged.
- **new_authzr_uri** (*str*) – Deprecated. Do not use.

Returns Authorization Resource.

Return type *AuthorizationResource*

Raises ***errors.WildcardUnsupportedError*** – if a wildcard is requested

request_issuance(*csr*, *authzrs*)
Request issuance.

Parameters

- **csr** (OpenSSL.crypto.X509Req wrapped in ComparableX509) – CSR
- **authzrs** – list of *AuthorizationResource*

Returns Issued certificate

Return type *messages.CertificateResource*

poll(*authzr*)
Poll Authorization Resource for status.

Parameters **authzr** (*AuthorizationResource*) – Authorization Resource

Returns Updated Authorization Resource and HTTP response.

Return type (*AuthorizationResource*, `requests.Response`)

poll_and_request_issuance(*csr*, *authzrs*, *mintime=5*, *max_attempts=10*)
Poll and request issuance.

This function polls all provided Authorization Resource URIs until all challenges are valid, respecting `Retry-After` HTTP headers, and then calls *request_issuance*.

Parameters

- **csr** (*ComparableX509*) – CSR (`OpenSSL.crypto.X509Req` wrapped in `ComparableX509`)
- **authzrs** – `list` of *AuthorizationResource*
- **mintime** (*int*) – Minimum time before next attempt, used if `Retry-After` is not present in the response.
- **max_attempts** (*int*) – Maximum number of attempts (per authorization) before `PollError` with non-empty `waiting` is raised.

Returns `(cert, updated_authzrs)` `tuple` where `cert` is the issued certificate (*messages.CertificateResource*), and `updated_authzrs` is a `tuple` consisting of updated Authorization Resources (*AuthorizationResource*) as present in the responses from server, and in the same order as the input `authzrs`.

Return type `tuple`

Raises ***PollError*** – in case of timeout or if some authorization was marked by the CA as invalid

check_cert(*certr*)
Check for new cert.

Parameters **certr** (*CertificateResource*) – Certificate Resource

Returns Updated Certificate Resource.

Return type *CertificateResource*

refresh(*certr*)
Refresh certificate.

Parameters **certr** (*CertificateResource*) – Certificate Resource

Returns Updated Certificate Resource.

Return type *CertificateResource*

fetch_chain(*certr*, *max_length=10*)
Fetch chain for certificate.

Parameters

- **certr** (CertificateResource) – Certificate Resource
- **max_length** (*int*) – Maximum allowed length of the chain. Note that each element in the certificate requires new `HTTP GET` request, and the length of the chain is controlled by the ACME CA.

Raises *errors.Error* – if recursion exceeds `max_length`

Returns Certificate chain for the Certificate Resource. It is a list ordered so that the first element is a signer of the certificate from Certificate Resource. Will be empty if `cert_chain_uri` is `None`.

Return type `list` of `OpenSSL.crypto.X509` wrapped in `ComparableX509`

revoke(*cert*, *rsn*)
Revoke certificate.

Parameters

- **cert** (*ComparableX509*) – `OpenSSL.crypto.X509` wrapped in `ComparableX509`
- **rsn** (*int*) – Reason code for certificate revocation.

Raises *ClientError* – If revocation is unsuccessful.

class acme.client.**ClientV2**(*directory*, *net*)
Bases: *acme.client.ClientBase*

ACME client for a v2 API.

Variables

- **directory** (messages.Directory) –
- **net** (ClientNetwork) – Client network.

new_account(*new_account*)
Register.

Parameters **new_account** (NewRegistration) –

Raises *ConflictError* – in case the account already exists

Returns Registration Resource.

Return type *RegistrationResource*

query_registration(*regr*)
Query server about registration.

Parameters **messages.RegistrationResource** – Existing Registration Resource.

update_registration(*regr*, *update=None*)
Update registration.

Parameters

- **regr** (messages.RegistrationResource) – Registration Resource.
- **update** (messages.Registration) – Updated body of the resource. If not provided, body will be taken from `regr`.

Returns Updated Registration Resource.

Return type *RegistrationResource*

new_order(*csr_pem*)
Request a new Order object from the server.

Parameters **csr_pem** (*str*) – A CSR in PEM format.

Returns The newly created order.

Return type *OrderResource*

poll(*authzr*)
Poll Authorization Resource for status.

Parameters **authzr** (*AuthorizationResource*) – Authorization Resource

Returns Updated Authorization Resource and HTTP response.

Return type (*AuthorizationResource*, requests.Response)

poll_and_finalize(*orderr*, *deadline=None*)
Poll authorizations and finalize the order.

If no deadline is provided, this method will timeout after 90 seconds.

Parameters

- **orderr** (messages.OrderResource) – order to finalize
- **deadline** (*datetime.datetime*) – when to stop polling and timeout

Returns finalized order

Return type *messages.OrderResource*

poll_authorizations(*orderr*, *deadline*)
Poll Order Resource for status.

finalize_order(*orderr*, *deadline*)
Finalize an order and obtain a certificate.

Parameters

- **orderr** (messages.OrderResource) – order to finalize
- **deadline** (*datetime.datetime*) – when to stop polling and timeout

Returns finalized order

Return type *messages.OrderResource*

revoke(*cert*, *rsn*)
Revoke certificate.

Parameters

- **cert** (*ComparableX509*) – OpenSSL.crypto.X509 wrapped in ComparableX509
- **rsn** (*int*) – Reason code for certificate revocation.

Raises ***ClientError*** – If revocation is unsuccessful.

external_account_required()
Checks if ACME server requires External Account Binding authentication.

class acme.client.**BackwardsCompatibleClientV2**(*net*, *key*, *server*)
Bases: object

ACME client wrapper that tends towards V2-style calls, but supports V1 servers.

Note: While this class handles the majority of the differences between versions of the ACME protocol, if you need to support an ACME server based on version 3 or older of the IETF ACME draft that uses combinations in authorizations (or lack thereof) to signal that the client needs to complete something other than any single challenge in the authorization to make it valid, the user of this class needs to understand and handle these

differences themselves. This does not apply to either of Let's Encrypt's endpoints where successfully completing any challenge in an authorization will make it valid.

Variables

- **acme_version** (*int*) – 1 or 2, corresponding to the Let's Encrypt endpoint
- ***client*** (ClientBase) – either Client or ClientV2

new_account_and_tos(*regr*, *check_tos_cb=None*)

Combined register and agree_tos for V1, new_account for V2

Parameters

- **regr** (NewRegistration) –
- **check_tos_cb** (*callable*) – callback that raises an error if the check does not work

new_order(*csr_pem*)

Request a new Order object from the server.

If using ACMEv1, returns a dummy OrderResource with only the authorizations field filled in.

Parameters **csr_pem** (*str*) – A CSR in PEM format.

Returns The newly created order.

Return type *OrderResource*

Raises ***errors.WildcardUnsupportedError*** – if a wildcard domain is requested but unsupported by the ACME version

finalize_order(*orderr*, *deadline*)

Finalize an order and obtain a certificate.

Parameters

- **orderr** (messages.OrderResource) – order to finalize
- **deadline** (*datetime.datetime*) – when to stop polling and timeout

Returns finalized order

Return type *messages.OrderResource*

revoke(*cert*, *rsn*)

Revoke certificate.

Parameters

- **cert** (*ComparableX509*) – OpenSSL.crypto.X509 wrapped in ComparableX509
- **rsn** (*int*) – Reason code for certificate revocation.

Raises ***ClientError*** – If revocation is unsuccessful.

external_account_required()

Checks if the server requires an external account for ACMEv2 servers.

Always return False for ACMEv1 servers, as it doesn't use External Account Binding.

class acme.client.**ClientNetwork**(*key*, *account=None*, *alg=RS256*, *verify_ssl=True*, *user_agent='acme-python'*, *timeout=45*, *source_address=None*)

Bases: object

Wrapper around requests that signs POSTs for authentication.

Also adds user agent, and handles Content-Type.

REPLAY_NONCE_HEADER = 'Replay-Nonce'

Initialize.

Parameters

- **key** (*josepy.JWK*) – Account private key
- **account** (messages.RegistrationResource) – Account object. Required if you are planning to use .post() with acme_version=2 for anything other than creating a new account; may be set later after registering.
- **alg** (*josepy.JWASignature*) – Algorithm to use in signing JWS.
- **verify_ssl** (*bool*) – Whether to verify certificates on SSL connections.
- **user_agent** (*str*) – String to send as User-Agent header.
- **timeout** (*float*) – Timeout for requests.
- **source_address** (*str or tuple(str, int)*) – Optional source address to bind to when making requests.

head(**args*, ***kwargs*)

Send HEAD request without checking the response.

Note, that _check_response is not called, as it is expected that status code other than successfully 2xx will be returned, or messages2.Error will be raised by the server.

get(*url*, *content_type='application/json'*, ***kwargs*)

Send GET request and check response.

post(**args*, ***kwargs*)

POST object wrapped in JWS and check response.

If the server responded with a badNonce error, the request will be retried once.

1.3 Errors

ACME errors.

exception acme.errors.**Error**

Bases: exceptions.Exception

Generic ACME error.

exception acme.errors.**DependencyError**

Bases: *acme.errors.Error*

Dependency error

exception acme.errors.**SchemaValidationError**

Bases: josepy.errors.DeserializationError

JSON schema ACME object validation error.

exception acme.errors.**ClientError**

Bases: *acme.errors.Error*

Network error.

exception acme.errors.**UnexpectedUpdate**
Bases: *acme.errors.ClientError*

Unexpected update error.

exception acme.errors.**NonceError**
Bases: *acme.errors.ClientError*

Server response nonce error.

exception acme.errors.**BadNonce**(*nonce*, *error*, **args*, ***kwargs*)
Bases: *acme.errors.NonceError*

Bad nonce error.

exception acme.errors.**MissingNonce**(*response*, **args*, ***kwargs*)
Bases: *acme.errors.NonceError*

Missing nonce error.

According to the specification an "ACME server MUST include an Replay-Nonce header field in each successful response to a POST it provides to a client (...)".

Variables ***response*** (*requests.Response*) – HTTP Response

exception acme.errors.**PollError**(*exhausted*, *updated*)
Bases: *acme.errors.ClientError*

Generic error when polling for authorization fails.

This might be caused by either timeout (`exhausted` will be non-empty) or by some authorization being invalid.

Variables

- **exhausted** – Set of *AuthorizationResource* that didn't finish within max allowed attempts.
- **updated** – Mapping from original *AuthorizationResource* to the most recently updated one

timeout
Was the error caused by timeout?

exception acme.errors.**ValidationError**(*failed_authzrs*)
Bases: *acme.errors.Error*

Error for authorization failures. Contains a list of authorization resources, each of which is invalid and should have an error field.

exception acme.errors.**TimeoutError**
Bases: *acme.errors.Error*

Error for when polling an authorization or an order times out.

exception acme.errors.**IssuanceError**(*error*)
Bases: *acme.errors.Error*

Error sent by the server after requesting issuance of a certificate.

exception acme.errors.**ConflictError**(*location*)
Bases: *acme.errors.ClientError*

Error for when the server returns a 409 (Conflict) HTTP status.

In the version of ACME implemented by Boulder, this is used to find an account if you only have the private key, but don't know the account URL.

Also used in V2 of the ACME client for the same purpose.

exception acme.errors.**WildcardUnsupportedError**
Bases: *acme.errors.Error*

Error for when a wildcard is requested but is unsupported by ACME CA.

1.4 Fields

ACME JSON fields.

class acme.fields.**Fixed**(*json_name*, *value*)
Bases: josepy.json_util.Field

Fixed field.

decode(*value*)
Decode a value, optionally with context JSON object.

encode(*value*)
Encode a value, optionally with context JSON object.

class acme.fields.**RFC3339Field**(*json_name*, *default=None*, *omitempty=False*, *decoder=None*, *encoder=None*)
Bases: josepy.json_util.Field

RFC3339 field encoder/decoder.

Handles decoding/encoding between RFC3339 strings and aware (not naive) datetime.datetime objects (e.g. datetime.datetime.now(pytz.utc)).

classmethod default_encoder(*value*)
Default (passthrough) encoder.

classmethod default_decoder(*value*)
Default decoder.

Recursively deserialize into immutable types (josepy.util.frozendict instead of dict(), tuple() instead of list()).

class acme.fields.**Resource**(*resource_type*, **args*, ***kwargs*)
Bases: josepy.json_util.Field

Resource MITM field.

decode(*value*)
Decode a value, optionally with context JSON object.

1.5 JOSE

The acme.jose module was moved to its own package "josepy". Please refer to its documentation there.

1.6 Messages

ACME protocol messages.

acme.messages.**is_acme_error**(*err*)
Check if argument is an ACME error.

exception acme.messages.**Error**(***kwargs*)
Bases: josepy.json_util.JSONObjectWithFields, *acme.errors.Error*

ACME error.

https://tools.ietf.org/html/draft-ietf-appsawg-http-problem-00

Variables

- **typ** (*unicode*) –
- **title** (*unicode*) –
- **detail** (*unicode*) –

classmethod with_code(*code*, ***kwargs*)
Create an Error instance with an ACME Error code.

Unicode code An ACME error code, like 'dnssec'.

Kwargs kwargs to pass to Error.

description
Hardcoded error description based on its type.

Returns Description if standard ACME error or None.

Return type unicode

code
ACME error code.

Basically self.typ without the ERROR_PREFIX.

Returns error code if standard ACME code or None.

Return type unicode

class acme.messages.**Status**(*name*)
Bases: acme.messages._Constant

ACME "status" field.

class acme.messages.**IdentifierType**(*name*)
Bases: acme.messages._Constant

ACME identifier type.

class acme.messages.**Identifier**(***kwargs*)
Bases: josepy.json_util.JSONObjectWithFields

ACME identifier.

Variables

- **typ** (IdentifierType) –
- **value** (*unicode*) –

class acme.messages.**Directory**(*jobj*)
Bases: josepy.interfaces.JSONDeSerializable

Directory.

class Meta(***kwargs*)
Bases: josepy.json_util.JSONObjectWithFields

Directory Meta.

terms_of_service
URL for the CA TOS

classmethod register(*resource_body_cls*)
Register resource.

to_partial_json()
Partially serialize.

Following the example, **partial serialization** means the following:

```
assert isinstance(Bar().to_partial_json()[0], Foo)
assert isinstance(Bar().to_partial_json()[1], Foo)

# in particular...
assert Bar().to_partial_json() != ['foo', 'foo']
```

Raises **josepy.errors.SerializationError** – in case of any serialization error.

Returns Partially serializable object.

classmethod from_json(*jobj*)
Deserialize a decoded JSON document.

Parameters **jobj** – Python object, composed of only other basic data types, as decoded from JSON document. Not necessarily dict (as decoded from "JSON object" document).

Raises **josepy.errors.DeserializationError** – if decoding was unsuccessful, e.g. in case of unparseable X509 certificate, or wrong padding in JOSE base64 encoded string, etc.

class acme.messages.**Resource**(***kwargs*)
Bases: josepy.json_util.JSONObjectWithFields

ACME Resource.

Variables **body** (acme.messages.ResourceBody) – Resource body.

class acme.messages.**ResourceWithURI**(***kwargs*)
Bases: *acme.messages.Resource*

ACME Resource with URI.

Variables ***uri*** (*unicode*) – Location of the resource.

class acme.messages.**ResourceBody**(***kwargs*)
Bases: josepy.json_util.JSONObjectWithFields

ACME Resource Body.

class acme.messages.**ExternalAccountBinding**
Bases: object

ACME External Account Binding

classmethod from_data(*account_public_key*, *kid*, *hmac_key*, *directory*)
Create External Account Binding Resource from contact details, kid and hmac.

class acme.messages.**Registration**(**kwargs)
Bases: *acme.messages.ResourceBody*

Registration Resource Body.

Variables

- **key** (*josepy.jwk.JWK*) – Public key.
- **contact** (*tuple*) – Contact information following ACME spec, `tuple` of `unicode`.
- **agreement** (*unicode*) –

classmethod from_data(*phone=None*, *email=None*, *external_account_binding=None*, **kwargs*)
Create registration resource from contact details.

phones
All phones found in the `contact` field.

emails
All emails found in the `contact` field.

class acme.messages.**NewRegistration**(**kwargs)
Bases: `acme.mixins.ResourceMixin`, *acme.messages.Registration*

New registration.

class acme.messages.**UpdateRegistration**(**kwargs)
Bases: `acme.mixins.ResourceMixin`, *acme.messages.Registration*

Update registration.

class acme.messages.**RegistrationResource**(**kwargs)
Bases: *acme.messages.ResourceWithURI*

Registration Resource.

Variables

- **body** (`acme.messages.Registration`) –
- **new_authzr_uri** (*unicode*) – Deprecated. Do not use.
- ***terms_of_service*** (*unicode*) – URL for the CA TOS.

class acme.messages.**ChallengeBody**(**kwargs)
Bases: *acme.messages.ResourceBody*

Challenge Resource Body.

Variables

- ***acme.challenges.Challenge*** – Wrapped challenge. Conveniently, all challenge fields are proxied, i.e. you can call `challb.x` to get `challb.chall.x` contents.
- **status** (`acme.messages.Status`) –
- **validated** (*datetime.datetime*) –
- **error** (`messages.Error`) –

encode(*name*)
Encode a single field.

Parameters **name** (*str*) – Name of the field to be encoded.

Raises

- **errors.SerializationError** – if field cannot be serialized
- ***errors.Error*** – if field could not be found

to_partial_json()
Partially serialize.

Following the example, **partial serialization** means the following:

```
assert isinstance(Bar().to_partial_json()[0], Foo)
assert isinstance(Bar().to_partial_json()[1], Foo)

# in particular...
assert Bar().to_partial_json() != ['foo', 'foo']
```

Raises **josepy.errors.SerializationError** – in case of any serialization error.

Returns Partially serializable object.

classmethod fields_from_json(*jobj*)
Deserialize fields from JSON.

uri
The URL of this challenge.

class acme.messages.**ChallengeResource**(***kwargs*)
Bases: *acme.messages.Resource*

Challenge Resource.

Variables

- **body** (acme.messages.ChallengeBody) –
- **authzr_uri** (*unicode*) – URI found in the 'up' Link header.

uri
The URL of the challenge body.

class acme.messages.**Authorization**(***kwargs*)
Bases: *acme.messages.ResourceBody*

Authorization Resource Body.

Variables

- **identifier** (acme.messages.Identifier) –
- ***challenges*** (*list*) – list of *ChallengeBody*
- **combinations** (*tuple*) – Challenge combinations (tuple of tuple of int, as opposed to list of list from the spec).
- **status** (acme.messages.Status) –
- **expires** (*datetime.datetime*) –

resolved_combinations
Combinations with challenges instead of indices.

class acme.messages.**NewAuthorization**(***kwargs*)
Bases: acme.mixins.ResourceMixin, *acme.messages.Authorization*

New authorization.

class acme.messages.**UpdateAuthorization**(***kwargs*)
Bases: acme.mixins.ResourceMixin, *acme.messages.Authorization*

Update authorization.

class acme.messages.**AuthorizationResource**(***kwargs*)
Bases: *acme.messages.ResourceWithURI*

Authorization Resource.

Variables

- **body** (acme.messages.Authorization) –
- **new_cert_uri** (*unicode*) – Deprecated. Do not use.

class acme.messages.**CertificateRequest**(***kwargs*)
Bases: acme.mixins.ResourceMixin, josepy.json_util.JSONObjectWithFields

ACME new-cert request.

Variables **csr** (*josepy.util.ComparableX509*) – OpenSSL.crypto.X509Req wrapped in ComparableX509

class acme.messages.**CertificateResource**(***kwargs*)
Bases: *acme.messages.ResourceWithURI*

Certificate Resource.

Variables

- **body** (*josepy.util.ComparableX509*) – OpenSSL.crypto.X509 wrapped in ComparableX509
- **cert_chain_uri** (*unicode*) – URI found in the 'up' Link header
- **authzrs** (*tuple*) – tuple of *AuthorizationResource*.

class acme.messages.**Revocation**(***kwargs*)
Bases: acme.mixins.ResourceMixin, josepy.json_util.JSONObjectWithFields

Revocation message.

Variables **certificate** (*ComparableX509*) – OpenSSL.crypto.X509 wrapped in ComparableX509

class acme.messages.**Order**(***kwargs*)
Bases: *acme.messages.ResourceBody*

Order Resource Body.

Variables

- **of .Identifier** (*list*) – List of identifiers for the certificate.
- **status** (acme.messages.Status) –
- **of str authorizations** (*list*) – URLs of authorizations.
- **certificate** (*str*) – URL to download certificate as a fullchain PEM.
- **finalize** (*str*) – URL to POST to to request issuance once all authorizations have "valid" status.
- **expires** (*datetime.datetime*) – When the order expires.
- **error** (Error) – Any error that occurred during finalization, if applicable.

class acme.messages.**OrderResource**(**kwargs)
Bases: *acme.messages.ResourceWithURI*

Order Resource.

Variables

- **body** (*acme.messages.Order*) –
- **csr_pem** (*str*) – The CSR this Order will be finalized with.
- **of acme.messages.AuthorizationResource authorizations** (*list*) – Fully-fetched AuthorizationResource objects.
- **fullchain_pem** (*str*) – The fetched contents of the certificate URL produced once the order was finalized, if it's present.

class acme.messages.**NewOrder**(**kwargs)
Bases: *acme.messages.Order*

New order.

1.7 Standalone

Support for standalone client challenge solvers.

class acme.standalone.**TLSServer**(*args, **kwargs)
Bases: SocketServer.TCPServer

Generic TLS Server.

class acme.standalone.**ACMEServerMixin**
ACME server common settings mixin.

class acme.standalone.**BaseDualNetworkedServers**(*ServerClass*, *server_address*, **remaining_args*, ***kwargs*)
Bases: object

Base class for a pair of IPv6 and IPv4 servers that tries to do everything it's asked for both servers, but where failures in one server don't affect the other.

If two servers are instantiated, they will serve on the same port.

serve_forever()
Wraps socketserver.TCPServer.serve_forever

getsocknames()
Wraps socketserver.TCPServer.socket.getsockname

shutdown_and_server_close()
Wraps socketserver.TCPServer.shutdown, socketserver.TCPServer.server_close, and threading.Thread.join

class acme.standalone.**TLSALPN01Server**(*server_address*, *certs*, *challenge_certs*, *ipv6=False*)
Bases: *acme.standalone.TLSServer*, *acme.standalone.ACMEServerMixin*

TLSALPN01 Server.

class acme.standalone.**HTTPServer**(*args, **kwargs)
Bases: BaseHTTPServer.HTTPServer

Generic HTTP Server.

class acme.standalone.**HTTP01Server**(*server_address*, *resources*, *ipv6=False*)
Bases: *acme.standalone.HTTPServer*, *acme.standalone.ACMEServerMixin*

HTTP01 Server.

class acme.standalone.**HTTP01DualNetworkedServers**(**args*, ***kwargs*)
Bases: *acme.standalone.BaseDualNetworkedServers*

HTTP01Server Wrapper. Tries everything for both. Failures for one don't affect the other.

class acme.standalone.**HTTP01RequestHandler**(**args*, ***kwargs*)
Bases: BaseHTTPServer.BaseHTTPRequestHandler

HTTP01 challenge handler.

Adheres to the stdlib's socketserver.BaseRequestHandler interface.

Variables **simple_http_resources** (*set*) – A set of *HTTP01Resource* objects. TODO: better name?

class **HTTP01Resource**(*chall*, *response*, *validation*)
Bases: tuple

chall
Alias for field number 0

response
Alias for field number 1

validation
Alias for field number 2

log_message(*format*, **args*)
Log arbitrary message.

handle()
Handle request.

handle_index()
Handle index page.

handle_404()
Handler 404 Not Found errors.

handle_simple_http_resource()
Handle HTTP01 provisioned resources.

classmethod **partial_init**(*simple_http_resources*)
Partially initialize this handler.

This is useful because socketserver.BaseServer takes uninitialized handler and initializes it with the current request.

ACME protocol implementation.

This module is an implementation of the ACME protocol.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W