CONTENTS

1 API Documentation 3
  1.1 Challenges .................................................. 3
  1.2 Client .......................................................... 9
  1.3 Errors .......................................................... 16
  1.4 Fields .......................................................... 18
  1.5 JOSE ............................................................ 18
  1.6 Messages ....................................................... 19
  1.7 Standalone ..................................................... 24

2 Indices and tables 27

Python Module Index 29

Index 31
Contents:
1.1 Challenges

ACME Identifier Validation Challenges.

class acme.challenges.Challenge(**kwargs: Any)
   ACME challenge.

   classmethod from_json(jobj: Mapping[str, Any]) → Union[acme.challenges.GenericChallenge, acme.challenges.UnrecognizedChallenge]
   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: Any)
   ACME challenge response.

class acme.challenges.UnrecognizedChallenge(jobj: Mapping[str, Any])
   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() → Dict[str, Any]
   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: Mapping[str, Any]) → acme.challenges.UnrecognizedChallenge
   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: Any)
   Response to Challenges based on Key Authorization.

   Parameters key_authorization (str) –

    verify(chall: acme.challenges.KeyAuthorizationChallenge, account_public_key: josepy.jwk.JWK) → bool
   Verify the key authorization.
Parameters

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

Returns: True if verification of the key authorization was successful.

Return type: `bool`

to_partial_json() → Dict[str, Any]

See josepy.JSONDeserializable.to_partial_json()

class acme.challenges.KeyAuthorizationChallenge(**kwargs: Any)

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: josepy.jwk.JWK) → str

Generate Key Authorization.

Parameters: account_key (`JWK`) –

Rtype: str

response(account_key: josepy.jwk.JWK) → acme.challenges.KeyAuthorizationChallengeResponse

Generate response to the challenge.

Parameters: account_key (`JWK`) –

Returns: Response (initialized response_cls) to the challenge.

Return type: `KeyAuthorizationChallengeResponse`

abstract validation(account_key: josepy.jwk.JWK, **kwargs: Any) → Any

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.


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: Any)

ACME dns-01 challenge response.

simple_verify(chall: acme.challenges.DNS01, domain: str, account_public_key: josepy.jwk.JWK) → bool

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

**Parameters**

- **chall** ([`challenges.DNS01`]) – Corresponding challenge.
- **domain** (`str`) – 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: Any)

ACME dns-01 challenge.

**response_cls**

alias of `acme.challenges.DNS01Response`

**LABEL = '_acme-challenge'**

Label clients prepend to the domain name being validated.

**validation(account_key: josepy.jwk.JWK, **unused_kwargs: Any) → str**

Generate validation.

**Parameters** account_key ([`JWK`]) –

**Return type** str

class acme.challenges.HTTP01Response(**kwargs: Any)

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: acme.challenges.HTTP01, domain: str, account_public_key: josepy.jwk.JWK, port: Optional[int] = None) → bool**

Simple verify.

**Parameters**

- **chall** ([`challenges.SimpleHTTP`]) – Corresponding challenge.
- **domain** (`str`) – 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: Any)
    ACME http-01 challenge.

    response_cls
        alias of acme.challenges.HTTP01Response

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

    property path: str
        Path (starting with '/') for provisioned resource.
    
        Return type str

    uri(domain: str) -> str
        Create an URI to the provisioned resource.
        
        Forms an URI to the HTTPS server provisioned resource (containing token).
    
        Parameters domain (str) -- Domain name being verified.
    
        Return type str

    validation(account_key: josepy.jwk.JWK, **unused_kwargs: Any) -> str
        Generate validation.
    
        Parameters account_key (JWK) --
    
        Return type str

class acme.challenges.TLSALPN01Response(**kwargs: Any)
    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.

    property h: bytes
        Hash value stored in challenge certificate

    gen_cert(domain: str, key: Optional[OpenSSL.crypto.PKey] = None, bits: int = 2048) -> Tuple[OpenSSL.crypto.X509, OpenSSL.crypto.PKey]
        Generate tls-alpn-01 certificate.
    
        Parameters
        • domain (str) -- 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: str, host: Optional[str] = None, port: Optional[int] = None) -> OpenSSL.crypto.X509
        Probe tls-alpn-01 challenge certificate.
    
        Parameters
        • domain (str) -- domain being validated, required.
        • host (str) -- IP address used to probe the certificate.
        • port (int) -- Port used to probe the certificate.
verify_cert(\textit{domain: str, cert: OpenSSL.crypto.X509}) \to \text{bool}

Verify tls-alpn-01 challenge certificate.

\textbf{Parameters}

\begin{itemize}
  \item \textit{domain (str)} – Domain name being validated.
  \item \textit{cert (OpenSSL.crypto.X509)} – Challenge certificate.
\end{itemize}

\textbf{Returns} Whether the certificate was successfully verified.

\textbf{Return type} \text{bool}

\begin{small}
\begin{verbatim}

simple_verify(chall: acme.challenges.TLSALPN01, domain: str, account_public_key: josepy.jwk.JWK, cert: Optional[OpenSSL.crypto.X509] = None, host: Optional[str] = None, port: Optional[int] = None) \to \text{bool}

Simple verify.

Verify validation using \textit{account_public_key}, optionally probe tls-alpn-01 certificate and check using \textit{verify_cert}.

\textbf{Parameters}

\begin{itemize}
  \item \textit{chall (challenges.TLSALPN01)} – Corresponding challenge.
  \item \textit{domain (str)} – Domain name being validated.
  \item \textit{account_public_key (JWK)} –
  \item \textit{cert (OpenSSL.crypto.X509)} – Optional certificate. If not provided (None) certificate will be retrieved using \textit{probe_cert}.
  \item \textit{host (string)} – IP address used to probe the certificate.
  \item \textit{port (int)} – Port used to probe the certificate.
\end{itemize}

\textbf{Returns} True if and only if client’s control of the domain has been verified.

\textbf{Return type} \text{bool}
\end{verbatim}
\end{small}

class acme.challenges.TLSALPN01(**kwargs: Any)

ACME tls-alpn-01 challenge.

\begin{small}
\begin{verbatim}

response_cls
  alias of acme.challenges.TLSALPN01Response

validation(account_key: josepy.jwk.JWK, **kwargs: Any) \to \text{Tuple[OpenSSL.crypto.X509, OpenSSL.crypto.PKey]}

Generate validation.

\textbf{Parameters}

\begin{itemize}
  \item \textit{account_key (JWK)} –
  \item \textit{domain (str)} – Domain verified by the challenge.
  \item \textit{cert_key (OpenSSL.crypto.PKey)} – Optional private key used in certificate generation. If not provided (None), then fresh key will be generated.
\end{itemize}

\textbf{Return type} \text{tuple of OpenSSL.crypto.X509 and OpenSSL.crypto.PKey}
\end{verbatim}
\end{small}

static is_supported() \to \text{bool}

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.

\textbf{Returns} True if TLS-ALPN-01 is supported on this machine, False otherwise.
class acme.challenges.DNS(**kwargs: Any)
ACME “dns” challenge.

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

gen_validation(account_key: josepy.jwk.JWK, alg: josepy.jwa.JWASignature = RS256, **kwargs: Any) → josepy.jws.JWS
Generate validation.

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

Returns  This challenge wrapped in JWS

Return type  JWS

check_validation(validation: josepy.jws.JWS, account_public_key: josepy.jwk.JWK) → bool
Check validation.

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

Return type  bool

gen_response(account_key: josepy.jwk.JWK, **kwargs: Any) → acme.challenges.DNSResponse
Generate response.

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

Return type  DNSResponse

validation_domain_name(name: str) → str
Domain name for TXT validation record.

Parameters  name (str) – Domain name being validated.

class acme.challenges.DNSResponse(**kwargs: Any)
ACME “dns” challenge response.

Parameters  validation (JWS) –

check_validation(chall: acme.challenges.DNS, account_public_key: josepy.jwk.JWK) → bool
Check validation.

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

Return type  bool
1.2 Client

Internal class delegating to a module, and displaying warnings when attributes related to deprecated attributes in the acme.client module.

```python
```

ACME client base object.

**Variables**

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

**update_registration(regr: acme.messages.RegistrationResource, update: Optional[acme.messages.Registration] = None) -> acme.messages.RegistrationResource**

Update registration.

**Parameters**

- `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: acme.messages.RegistrationResource) -> acme.messages.RegistrationResource**

Deactivate registration.

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

**Returns** The Registration resource that was deactivated.

**Return type RegistrationResource**


Deactivate authorization.


**Returns** The Authorization resource that was deactivated.

**Return type AuthorizationResource**


Answer challenge.

**Parameters**

- `challb` (ChallengeBody) – Challenge Resource body.
Returns  Challenge Resource with updated body.

Return type  ChallengeResource

Raises  UnexpectedUpdate –

classmethod  retry_after( response: requests.models.Response, default: int ) → datetime.datetime

Compute next poll time based on response Retry-After header.

Handles integers and various datenstring 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


ACME client for a v1 API.

Deprecated since version 1.18.0: Use ClientV2 instead.

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: Optional[acme.messages.NewRegistration] = None ) → acme.messages.RegistrationResource

Register.

Parameters new_reg (NewRegistration) –

Returns  Registration Resource.

Return type  RegistrationResource

query_registration( regr: acme.messages.RegistrationResource ) → acme.messages.RegistrationResource

Query server about registration.

Parameters regr (messages.RegistrationResource) – Existing Registration Resource.

agree_to_tos( regr: acme.messages.RegistrationResource ) → acme.messages.RegistrationResource

Agree to the terms-of-service.

Parameters regr (RegistrationResource) – Registration Resource.

Returns  Updated Registration Resource.

Return type  RegistrationResource
request_challenges(identifier: acme.messages.Identifier, new_authzr_uri: Optional[str] = None) → acme.messages.AuthorizationResource

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: str, new_authzr_uri: Optional[str] = None) → acme.messages.AuthorizationResource

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.

Parameters

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

Returns Issued certificate

Return type messages.CertificateResource


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

Returns Updated Authorization Resource and HTTP response.

Return type (AuthorizationResource, requests.Response)


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 for new cert.

Parameters certr (CertificateResource) – Certificate Resource

Returns Updated Certificate Resource.

Return type CertificateResource

Refresh certificate.

Parameters certr (CertificateResource) – Certificate Resource

Returns Updated Certificate Resource.

Return type CertificateResource

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: josepy.util.ComparableX509, rsn: int) → None
Revoke certificate.
Parameters

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

Raises `ClientError` – If revocation is unsuccessful.

ACME client for a v2 API.

Variables

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

new_account(new_account: acme.messages.NewRegistration) → acme.messages.RegistrationResource
Register.

Parameters `new_account` (*NewRegistration*) –

Raises `ConflictError` – in case the account already exists

Returns Registration Resource.

Return type `RegistrationResource`

query_registration(regr: acme.messages.RegistrationResource) → acme.messages.RegistrationResource
Query server about registration.

Parameters `regr` (*messages.RegistrationResource*) – Existing Registration Resource.

update_registration(regr: acme.messages.RegistrationResource, update: Optional[acme.messages.Registration] = None) → acme.messages.RegistrationResource
Update registration.

Parameters

- `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: bytes) → acme.messages.OrderResource
Request a new Order object from the server.

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

Returns The newly created order.

Return type `OrderResource`


Returns Updated Authorization Resource and HTTP response.

Return type (`AuthorizationResource`, `requests.Response`)
**Poll and Finalize**

```
```

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**

```
```

Poll Order Resource for status.

---

**Finalize Order**

```
```

Finalize an order and obtain a certificate.

**Parameters**

- **orderr** ([messages.OrderResource](#)) – order to finalize
- **deadline** ([datetime.datetime](#)) – when to stop polling and timeout
- **fetch_alternative_chains** ([bool](#)) – whether to also fetch alternative certificate chains

**Returns**

finalized order

**Return type**

[messages.OrderResource](#)

---

**Revoke**

```
revoke(cert: josepy.util.ComparableX509, rsn: int) → None
```

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**

```
external_account_required() → bool
```

Checks if ACME server requires External Account Binding authentication.

---

**Class**

```
```

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

Deprecated since version 1.18.0: Use `ClientV2` instead.

**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: acme.messages.NewRegistration, check_tos_cb: Optional[Callable[[str], None]] = None] -> acme.messages.RegistrationResource)

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: bytes) -> acme.messages.OrderResource

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` (bytes) – 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 an order and obtain a certificate.

Parameters

- `orderr` (messages.OrderResource) – order to finalize

- `deadline` (datetime.datetime) – when to stop polling and timeout

- `fetch_alternative_chains` (bool) – whether to also fetch alternative certificate chains

Returns finalized order

Return type messages.OrderResource

`revoke` (cert: josepy.util.ComparableX509, rsn: int) -> None

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` () -> bool

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: josepy.jwk.JWK
    account: Optional[acme.messages.RegistrationResource] = None
    alg: josepy.jwa.JWASignature = RS256
    verify_ssl: bool = True
    user_agent: str = 'acme-python'
    timeout: int = 45
    source_address: Optional[Union[str, Tuple[str, int]]] = None

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: Any, **kwargs: Any) -> requests.models.Response

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: str, content_type: str = 'application/json', **kwargs: Any) -> requests.models.Response

Send GET request and check response.

post(*args: Any, **kwargs: Any) -> requests.models.Response

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
    Generic ACME error.

exception acme.errors.DependencyError
    Dependency error

exception acme.errors.SchemaValidationError
    JSON schema ACME object validation error.

exception acme.errors.ClientError
    Network error.
exception acme.errors.UnexpectedUpdate
    Unexpected update error.

exception acme.errors.NonceError
    Server response nonce error.

exception acme.errors.BadNonce(nonce: str, error: Exception, *args: Any)
    Bad nonce error.

    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: Set[messages.AuthorizationResource], updated:
    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

    property timeout: bool
    Was the error caused by timeout?

exception acme.errors.ValidationError(failed_authzrs: List[messages.AuthorizationResource])
    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
    Error for when polling an authorization or an order times out.

exception acme.errors.IssuanceError(error: messages.Error)
    Error sent by the server after requesting issuance of a certificate.

exception acme.errors.ConflictError(location: str)
    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
    Error for when a wildcard is requested but is unsupported by ACME CA.
1.4 Fields

ACME JSON fields.

```python
class acme.fields.Fixed(json_name: str, value: Any)
    Fixed field.
    decode(value: Any) -> Any
        Decode a value, optionally with context JSON object.
    encode(value: Any) -> Any
        Encode a value, optionally with context JSON object.
```

```python
class acme.fields.RFC3339Field(json_name: str, default: Optional[Any] = None, omitempty: bool = False, decoder: Optional[Callable[[Any], Any]] = None, encoder: Optional[Callable[[Any], Any]] = None)
    RFC339 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: datetime.datetime) -> str
        Default (passthrough) encoder.
    classmethod default_decoder(value: str) -> datetime.datetime
        Default decoder.
        Recursively deserialize into immutable types (`josepy.util.frozendict` instead of `dict`, `tuple` instead of `list`).
```

```python
class acme.fields.Resource(resource_type: str, *args: Any, **kwargs: Any)
    Resource MITM field.
    decode(value: Any) -> Any
        Decode a value, optionally with context JSON object.
```

```python
acme.fields.fixed(json_name: str, value: Any) -> Any
    Generates a type-friendly Fixed field.
acme.fields.rfc3339(json_name: str, omitempty: bool = False) -> Any
    Generates a type-friendly RFC3339 field.
acme.fields.resource(resource_type: str) -> Any
    Generates a type-friendly Resource field.
```

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.

```python
acme.messages.is_acme_error(err: BaseException) → bool
```

Check if argument is an ACME error.

```python
exception acme.messages.Error(**kwargs: Any)
```

ACME error.


Variables

- **typ**(str)
- **title**(str)
- **detail**(str)

```python
classmethod with_code(code: str, **kwargs: Any) → acme.messages.Error
```

Create an Error instance with an ACME Error code.

- **code**(str)
  - An ACME error code, like 'dnssec'.
- **kwargs**
  - kwargs to pass to Error.

```python
property description: Optional[str]
```

Hardcoded error description based on its type.

- **Returns**
  - Description if standard ACME error or None.

```python
property code: Optional[str]
```

ACME error code.

- **Returns**
  - error code if standard ACME code or None.

```python
class acme.messages.Status(name: str)
```

ACME “status” field.

```python
class acme.messages.IdentifierType(name: str)
```

ACME identifier type.

```python
class acme.messages.Identifier(**kwargs: Any)
```

ACME identifier.

- **Variables**
  - **typ**(IdentifierType)
  - **value**(str)

```python
class acme.messages.HasResourceType
```

Represents a class with a resource_type class parameter of type string.

```python
class acme.messages.Directory(jobj: Mapping[str, Any])
```

Directory.

```python
class Meta(**kwargs: Any)
```

Directory Meta.
property terms_of_service: str

URL for the CA TOS

classmethod register(resource_body_cls: Type[acme.messages.GenericHasResourceType]) → Type[acme.messages.GenericHasResourceType]

Register resource.

to_partial_json() → Dict[str, Any]

Partially serialize.

 Following the example, partial serialization means the following:

```python
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']
```


Returns Partially serializable object.

classmethod from_json(jobj: MutableMapping[str, Any]) → acme.messages.Directory

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 unparsable X509 certificate, or wrong padding in JOSE base64 encoded string, etc.

class acme.messages.Resource(**kwargs: Any)

ACME Resource.

Variables

• body (acme.messages.ResourceBody) – Resource body.

class acme.messages.ResourceWithURI(**kwargs: Any)

ACME Resource with URI.

Variables

• uri (str) – Location of the resource.

class acme.messages.ResourceBody(**kwargs: Any)

ACME Resource Body.

class acme.messages.ExternalAccountBinding

ACME External Account Binding


Create External Account Binding Resource from contact details, kid and hmac.

class acme.messages.Registration(**kwargs: Any)

Registration Resource Body.

Variables

• key (jose.JWK) – Public key.

• contact (tuple) – Contact information following ACME spec, tuple of str.

• agreement (str) –
classmethod from_data(phone: Optional[str] = None, email: Optional[str] = None,
                   external_account_binding: Optional[Dict[str, Any]] = None, **kwargs: Any) ->
                   acme.messages.GenericRegistration

Create registration resource from contact details.

The contact keyword being passed to a Registration object is meaningful, so this function represents
empty iterables in its kwargs by passing on an empty tuple.

to_partial_json() -> Dict[str, Any]

Modify josepy.JSONDeserializable.to_partial_json()

fields_to_partial_json() -> Dict[str, Any]

Modify josepy.JSONObjectWithFields.fields_to_partial_json()

property phones: Tuple[str, ...]

All phones found in the contact field.

property emails: Tuple[str, ...]

All emails found in the contact field.

class acme.messages.NewRegistration(**kwargs: Any)

New registration.

class acme.messages.UpdateRegistration(**kwargs: Any)

Update registration.

class acme.messages.RegistrationResource(**kwargs: Any)

Registration Resource.

Variables

• body (acme.messages.Registration) –
• new_authzr_uri (str) – Deprecated. Do not use.
• terms_of_service (str) – URL for the CA TOS.

class acme.messages.ChallengeBody(**kwargs: Any)

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: str) -> Any

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() -> Dict[str, Any]

Partially serialize.

Following the example, partial serialization means the following:
```python
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']
```


Returns Partially serializable object.

classmethod `fields_from_json()`

Deserialize fields from JSON.

```python
property uri: str
The URL of this challenge.
```

class `acme.messages.ChallengeResource(**kwargs: Any)`

Challenge Resource.

Variables

- `body (acme.messages.ChallengeBody) –`
- `authzr_uri (str) –` URI found in the ‘up’ Link header.

```python
property uri: str
The URL of this challenge body.
```

class `acme.messages.Authorization(**kwargs: Any)`


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) –`

```python
property resolved_combinations: Tuple[Tuple[acme.messages.ChallengeBody, ...], ...]
Combinations with challenges instead of indices.
```

class `acme.messages.NewAuthorization(**kwargs: Any)`

New authorization.

class `acme.messages.UpdateAuthorization(**kwargs: Any)`

Update authorization.

class `acme.messages.AuthorizationResource(**kwargs: Any)`

Authorization Resource.

Variables

- `body (acme.messages.Authorization) –`
- `new_cert_uri (str) –` Deprecated. Do not use.

class `acme.messages.CertificateRequest(**kwargs: Any)`

ACME new-cert request.
Variables `csr` ([jose.ComparableX509](https://josepy.readthedocs.io/en/latest/josepy.html#josepy.ComparableX509)) – OpenSSL.crypto.X509Req wrapped in ComparableX509

```python
class acme.messages.CertificateResource(**kwargs: Any)
```

Certificate Resource.

Variables

- `cert_chain_uri` (str) – URI found in the ‘up’ Link header

```python
class acme.messages.Revocation(**kwargs: Any)
```

Revocation message.


```python
class acme.messages.Order(**kwargs: Any)
```

Order Resource Body.

Variables

- `identifiers` (list of [Identifier](https://acme-python.readthedocs.io/en/latest/acme.html#acme.messages.Identifier)) – List of identifiers for the certificate.
- `authorizations` (list of str) – URLs of authorizations.
- `certificate` (str) – URL to download certificate as a fullchain PEM.
- `finalize` (str) – URL to POST to request issuance once all authorizations have “valid” status.
- `expires` ([datetime.datetime](https://docs.python.org/3/library/datetime.html#datetime.datetime)) – When the order expires.
- `error` ([Error](https://acme-python.readthedocs.io/en/latest/acme.html#acme.Error)) – Any error that occurred during finalization, if applicable.

```python
class acme.messages.OrderResource(**kwargs: Any)
```

Order Resource.

Variables

- `body` ([acme.messages.Order](https://acme-python.readthedocs.io/en/latest/acme.html#acme.messages.Order)) –
- `csr_pem` (bytes) – The CSR this Order will be finalized with.
- `fullchain_pem` (str) – The fetched contents of the certificate URL produced once the order was finalized, if it's present.
- `alternative_fullchains_pem` (list of str) – The fetched contents of alternative certificate chain URLs produced once the order was finalized, if present and requested during finalization.

```python
class acme.messages.NewOrder(**kwargs: Any)
```

New order.
1.7 Standalone

Support for standalone client challenge solvers.

class acme.standalone.TLSServer(*args: Any, **kwargs: Any)
    Generic TLS Server.
        server_bind() → None
            Called by constructor to bind the socket.
            May be overridden.

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

class acme.standalone.BaseDualNetworkedServers(ServerClass: Type[socketserver.TCPServer],
    server_address: Tuple[str, int], *remaining_args: Any,
    **kwargs: Any)
    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() → None
            Wraps socketserver.TCPServer.serve_forever

        getsocknames() → List[Tuple[str, int]]
            Wraps socketserver.TCPServer.socket.getsockname

        shutdown_and_server_close() → None
            Wraps socketserver.TCPServer.shutdown, socketserver.TCPServer.server_close, and threading.Thread.join


class acme.standalone.TLSALPN01Server(server_address: Tuple[str, int], certs: List[Tuple[OpenSSL.crypto.PKey, OpenSSL.crypto.X509]], challenge_certs: Mapping[str, Tuple[OpenSSL.crypto.PKey, OpenSSL.crypto.X509]], ipv6: bool = False)
    TLSALPN01 Server.

class acme.standalone.HTTPServer(*args: Any, **kwargs: Any)
    Generic HTTP Server.

class acme.standalone.HTTP01Server(server_address: Tuple[str, int], resources: Set[acme.challenges.HTTP01], ipv6: bool = False, timeout: int = 30)
    HTTP01 Server.

class acme.standalone.HTTP01DualNetworkedServers(*args: Any, **kwargs: Any)
    HTTP01 Server Wrapper. Tries everything for both. Failures for one don’t affect the other.

class acme.standalone.HTTP01RequestHandler(*args: Any, **kwargs: Any)
    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)

            property chall
                Alias for field number 0
property response
    Alias for field number 1

property validation
    Alias for field number 2

property timeout: int
    The default timeout this server should apply to requests.  
    :return: timeout to apply  
    :rtype: int

log_message(format: str, *args: Any) → None
    Log arbitrary message.

handle() → None
    Handle request.

handle_index() → None
    Handle index page.

handle_404() → None
    Handler 404 Not Found errors.

handle_simple_http_resource() → None
    Handle HTTP01 provisioned resources.

classmethod partial_init(simple_http_resources: Set[acme.challenges.HTTP01], timeout: int) → functools.partial[HTTP01RequestHandler]
    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.
INDICES AND TABLES

- genindex
- modindex
- search
a
acme, 25
acme.challenges, 3
acme.client, 9
acme.errors, 16
acme.fields, 18
acme.messages, 19
acme.standalone, 24
<table>
<thead>
<tr>
<th>Module/Class</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>acme module</td>
<td>25</td>
</tr>
<tr>
<td>acme.challenges module</td>
<td>3</td>
</tr>
<tr>
<td>acme.client module</td>
<td>9</td>
</tr>
<tr>
<td>acme.errors module</td>
<td>16</td>
</tr>
<tr>
<td>acme.fields module</td>
<td>18</td>
</tr>
<tr>
<td>acme.messages module</td>
<td>19</td>
</tr>
<tr>
<td>acme.standalone module</td>
<td>24</td>
</tr>
<tr>
<td>ACMEServerMixin</td>
<td>(class in acme.standalone)</td>
</tr>
<tr>
<td>agree_to_tos()</td>
<td>(acme.client.Client method)</td>
</tr>
<tr>
<td>answer_challenge()</td>
<td>(acme.client.ClientBase method)</td>
</tr>
<tr>
<td>Authorization</td>
<td>(class in acme.messages)</td>
</tr>
<tr>
<td>AuthorizationResource</td>
<td>(class in acme.messages)</td>
</tr>
<tr>
<td>BackwardsCompatibleClientV2</td>
<td>(class in acme.client)</td>
</tr>
<tr>
<td>BadNonce</td>
<td>17</td>
</tr>
<tr>
<td>BaseDualNetworkedServers</td>
<td>(class in acme.standalone)</td>
</tr>
<tr>
<td>CertificateRequest</td>
<td>(class in acme.messages)</td>
</tr>
<tr>
<td>CertificateResource</td>
<td>(class in acme.messages)</td>
</tr>
<tr>
<td>chall</td>
<td>(acme.standalone.HTTP01RequestHandler.HTTP01Resource property)</td>
</tr>
<tr>
<td>Challenge</td>
<td>(class in acme.challenges)</td>
</tr>
<tr>
<td>ChallengeBody</td>
<td>(class in acme.messages)</td>
</tr>
<tr>
<td>ChallengeResource</td>
<td>(class in acme.messages)</td>
</tr>
<tr>
<td>ChallengeResponse</td>
<td>(class in acme.challenges)</td>
</tr>
<tr>
<td>check_cert()</td>
<td>(acme.client.Client method)</td>
</tr>
<tr>
<td>check_validation()</td>
<td>(acme.challenges.DNSResponse method)</td>
</tr>
<tr>
<td>Client</td>
<td>(class in acme.client)</td>
</tr>
<tr>
<td>ClientBase</td>
<td>(class in acme.client)</td>
</tr>
<tr>
<td>ClientError</td>
<td>16</td>
</tr>
<tr>
<td>ClientNetwork</td>
<td>(class in acme.client)</td>
</tr>
<tr>
<td>ClientV2</td>
<td>(class in acme.client)</td>
</tr>
<tr>
<td>code</td>
<td>(acme.messages.Error property)</td>
</tr>
<tr>
<td>ConflictError</td>
<td>17</td>
</tr>
<tr>
<td>deactivate_authorization()</td>
<td>(acme.client.ClientBase method)</td>
</tr>
<tr>
<td>deactivate_registration()</td>
<td>(acme.client.ClientBase method)</td>
</tr>
<tr>
<td>decode()</td>
<td>(acme.fields.Fixed method)</td>
</tr>
<tr>
<td>decode()</td>
<td>(acme.fields.Resource method)</td>
</tr>
<tr>
<td>default_decoder()</td>
<td>(acme.fields.RFC3339Field class method)</td>
</tr>
<tr>
<td>default_encoder()</td>
<td>(acme.fields.RFC3339Field class method)</td>
</tr>
<tr>
<td>DependencyError</td>
<td>16</td>
</tr>
<tr>
<td>description</td>
<td>(acme.messages.Error property)</td>
</tr>
<tr>
<td>Directory</td>
<td>(class in acme.messages)</td>
</tr>
<tr>
<td>Directory.Meta</td>
<td>(class in acme.messages)</td>
</tr>
<tr>
<td>DNS</td>
<td>(class in acme.challenges)</td>
</tr>
<tr>
<td>DNS01</td>
<td>(class in acme.challenges)</td>
</tr>
<tr>
<td>DNS01Response</td>
<td>(class in acme.challenges)</td>
</tr>
<tr>
<td>DNSResponse</td>
<td>(class in acme.challenges)</td>
</tr>
<tr>
<td>emails</td>
<td>(acme.messages.Registration property)</td>
</tr>
<tr>
<td>encode()</td>
<td>(acme.fields.Fixed method)</td>
</tr>
<tr>
<td>encode()</td>
<td>(acme.messages.ChallengeBody method)</td>
</tr>
<tr>
<td>Error</td>
<td>16, 19</td>
</tr>
<tr>
<td>external_account_required()</td>
<td>(acme.client.BackwardsCompatibleClientV2 method)</td>
</tr>
<tr>
<td>external_account_required()</td>
<td>(acme.client.ClientV2 method)</td>
</tr>
<tr>
<td>ExternalAccountBinding</td>
<td>(class in acme.messages)</td>
</tr>
</tbody>
</table>
acme-python Documentation, Release 0

F

fetch_chain() (acme.client.Client method), 12
fields_from_json() (acme.messages.ChallengeBody class method), 22
fields_to_partial_json() (acme.messages.Registration method), 21
finalize_order() (acme.client.BackwardsCompatibleClient method), 15
finalize_order() (acme.client.ClientV2 method), 14
Fixed (class in acme.fields), 18
fixed() (in module acme.fields), 18
from_data() (acme.messages.ExternalAccountBinding class method), 20
from_data() (acme.messages.Registration class method), 20
from_json() (acme.challenges.Challenge class method), 3
from_json() (acme.challenges.UnrecognizedChallenge class method), 3
from_json() (acme.messages.Directory class method), 20

G

gen_cert() (acme.challenges.TLSALPN01Response method), 6
gen_response() (acme.challenges.DNS method), 8
gen_validation() (acme.challenges.DNS method), 8
get() (acme.client.ClientNetwork method), 16
getsocknames() (acme.standalone.BaseDualNetworkedServers method), 24

H

h (acme.challenges.TLSALPN01Response property), 6
handle() (acme.standalone.HTTP01RequestHandler method), 25
handle_404() (acme.standalone.HTTP01RequestHandler method), 25
handle_index() (acme.standalone.HTTP01RequestHandler method), 25
handle_simple_http_resource() (acme.standalone.HTTP01RequestHandler method), 25
HasResourceType (class in acme.messages), 19
head() (acme.client.ClientNetwork method), 16
HTTP01 (class in acme.challenges), 5
HTTP01DualNetworkedServers (class in acme.standalone), 24
HTTP01RequestHandler (class in acme.standalone), 24
HTTP01RequestHandler.HTTP01Resource (class in acme.standalone), 24
HTTP01Response (class in acme.challenges), 5
HTTP01Server (class in acme.standalone), 24
HTTPServer (class in acme.standalone), 24

I

Identifier (class in acme.messages), 19
IdentifierType (class in acme.messages), 19
is_acme_error() (in module acme.messages), 19
is_supported() (acme.challenges.TLSALPN01 static method), 7
IssuanceError, 17

K

key_authorization() (acme.challenges.KeyAuthorizationChallenge method), 4
KeyAuthorizationChallenge (class in acme.challenges), 4
KeyAuthorizationChallengeResponse (class in acme.challenges), 3

L

LABEL (acme.challenges.DNS attribute), 8
LABEL (acme.challenges.DNS01 attribute), 5
tojson() (acme.standalone.HTTP01RequestHandler method), 25

M

MissingNonce, 17
module

acme, 25
acme.challenges, 3
acme.client, 9
acme.errors, 16
acme.fields, 18
acme.messages, 19
acme.standalone, 24

N

new_account() (acme.client.ClientV2 method), 13
new_account_and_tos() (acme.client.BackwardsCompatibleClientV2 method), 15
new_order() (acme.client.BackwardsCompatibleClientV2 method), 15
new_order() (acme.client.ClientV2 method), 13
NewAuthorization (class in acme.messages), 22
NewOrder (class in acme.messages), 23
NewRegistration (class in acme.messages), 21
NonceError, 17

O

Order (class in acme.messages), 23
OrderResource (class in acme.messages), 23

P

partial_init() (acme.standalone.HTTP01RequestHandler class method), 25
Index
UnrecognizedChallenge (class in acme.challenges), 3
update_registration() (acme.client.ClientBase method), 9
update_registration() (acme.client.ClientV2 method), 13
UpdateAuthorization (class in acme.messages), 22
UpdateRegistration (class in acme.messages), 21
uri (acme.messages.ChallengeBody property), 22
uri (acme.messages.ChallengeResource property), 22
uri() (acme.challenges.HTTP01 method), 6
URI_ROOT_PATH (acme.challenges.HTTP01 attribute), 6

V
validation (acme.standalone.HTTP01RequestHandler.HTTP01Resource property), 25
validation() (acme.challenges.DNS01 method), 5
validation() (acme.challenges.HTTP01 method), 6
validation() (acme.challenges.KeyAuthorizationChallenge method), 4
validation() (acme.challenges.TLSALPN01 method), 7
validation_domain_name() (acme.challenges.DNS method), 8
validation_domain_name() (acme.challenges.DNS01 method), 5
ValidationWarning, 17
verify() (acme.challenges.KeyAuthorizationChallengeResponse method), 3
verify_cert() (acme.challenges.TLSALPN01Response method), 6

W
WHITESPACE_CUTSET (acme.challenges.HTTP01Response attribute), 5
WildcardUnsupportedError, 17
with_code() (acme.messages.Error class method), 19