## CONTENTS

1 API Documentation  
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
1 Challenges

ACME Identifier Validation Challenges.

```python
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.**

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

Parameters
- account_key (JWK) –

**Return type** str
```

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

**Parameters**
- name (str) – Domain name being validated.

**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`.

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

```python
def 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**(*domain: str, cert: OpenSSL.crypto.X509*) → bool

Verify tls-alpn-01 challenge certificate.

**Parameters**

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

**Returns**

Whether the certificate was successfully verified.

**Return type** bool


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: Any**)

ACME tls-alpn-01 challenge.

**response_cls**

alias of *acme.challenges.TLSALPN01Response*

**validation**(*account_key: josepy.jwk.JWK, **kwargs: Any*) → Tuple[OpenSSL.crypto.X509, OpenSSL.crypto.PKey]

Generate validation.

**Parameters**

- **account_key** (*JWK*) –
- **domain** (*str*) – 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**() → 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.

**Returns**

True if TLS-ALPN-01 is supported on this machine, False otherwise.
Return type  bool

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.

```python
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

```python
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

```python
```

Deactivate authorization.


Returns The Authorization resource that was deactivated.

Return type AuthorizationResource

```python
```

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: requests.models.Response, default: int) → datetime.datetime

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


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.

**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 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**(*certr*: acme.messages.CertificateResource, **max_length**: int = 10) → List[josepy.util.ComparableX509]
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

- **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: 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**


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

Returns Updated Authorization Resource and HTTP response.


Poll authorizations and finalize the order.

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

Parameters

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

Returns finalized order

Return type messages.OrderResource


Poll Order Resource for status.


Finalize an order and obtain a certificate.

Parameters

• order (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 ACME server requires External Account Binding authentication.


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

Depreciated 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** *(acme.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 **acme.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.

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()).

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.
```

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

ACME identifier type.

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

ACME identifier.

**Variables**

- `typ (IdentifierType)`
- `value (str)`

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

ACME error.


**Variables**

- `typ (str)`
- `title (str)`
- `detail (str)`
- `identifier (Identifier)`
- `subproblems (tuple)` — An array of ACME Errors which may be present when the CA returns multiple errors related to the same request, tuple of `Error`.

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

Create an Error instance with an ACME Error code.

**Str code** 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.

**Return type** str

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

ACME error code.

Basically self.typ without the ERROR_PREFIX.

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

**Return type** str

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

ACME “status” field.

```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.

    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.
```

```python
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)*

```python
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*.

```python
to_partial_json() → Dict[str, Any]
```
Modify josepy.JSONDeserializable.to_partial_json()

```python
fields_to_partial_json() → Dict[str, Any]
```
Modify josepy.JSONObjectWithFields.fields_to_partial_json()

```python
property phones: Tuple[str, ...]
```
All phones found in the contact field.

```python
property emails: Tuple[str, ...]
```
All emails found in the contact field.

```python
class acme.messages.NewRegistration(**kwargs: Any)
```
New registration.

```python
class acme.messages.UpdateRegistration(**kwargs: Any)
```
Update registration.

```python
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.

```python
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)* –

```python
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() \rightarrow \text{Dict}[\text{str}, \text{Any}]

Partially serialize.

Following the example, \textit{partial serialization} means the following:

\begin{verbatim}
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']
\end{verbatim}

\textbf{Raises} \texttt{josepy.errors.SerializationError} – in case of any serialization error.

\textbf{Returns} Partially serializable object.

classmethod \texttt{fields_from_json}(\textit{jobj: Mapping[str, Any]} \rightarrow \text{Dict[str, Any]}

Deserializes fields from JSON.

property \texttt{uri}: \texttt{str}

The URL of this challenge.

class \texttt{acme.messages.ChallengeResource}(**\texttt{kwargs: Any})

Challenge Resource.

Variables

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

property \texttt{uri}: \texttt{str}

The URL of the challenge body.

class \texttt{acme.messages.Authorization}(**\texttt{kwargs: Any})


Variables

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

property \texttt{resolved_combinations}: \texttt{Tuple[Tuple[acme.messages.ChallengeBody, ...], ...]}

Combinations with challenges instead of indices.

class \texttt{acme.messages.NewAuthorization}(**\texttt{kwargs: Any})

New authorization.

class \texttt{acme.messages.UpdateAuthorization}(**\texttt{kwargs: Any})

Update authorization.

class \texttt{acme.messages.AuthorizationResource}(**\texttt{kwargs: Any})

Authorization Resource.

Variables

- \texttt{body} (\texttt{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) – OpenSSL.crypto.X509Req wrapped in ComparableX509

class acme.messages.CertificateResource(**kwargs: Any)
    Certificate Resource.
    Variables
    • `body (josepy.util.ComparableX509) – OpenSSL.crypto.X509 wrapped in ComparableX509`
    • `cert_chain_uri (str)` – URI found in the ‘up’ Link header
    • `authzrs (tuple)` – tuple of AuthorizationResource.

class acme.messages.Revocation(**kwargs: Any)
    Revocation message.
    Variables
certificate (jose.ComparableX509) – OpenSSL.crypto.X509 wrapped in jose.ComparableX509

class acme.messages.Order(**kwargs: Any)
    Order Resource Body.
    Variables
    • `identifiers (list of Identifier)` – List of identifiers for the certificate.
    • `status (acme.messages.Status)` –
    • `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)` – When the order expires.
    • `error (Error)` – Any error that occurred during finalization, if applicable.

class acme.messages.OrderResource(**kwargs: Any)
    Order Resource.
    Variables
    • `body (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.

class acme.messages.NewOrder(**kwargs: Any)
    New order.
### 1.7 Standalone

Support for standalone client challenge solvers.

```python
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)
    HTTP01Server 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
PYTHON MODULE INDEX

a
acme, 25
acme.challenges, 3
acme.client, 9
acme.errors, 16
acme.fields, 18
acme.messages, 19
acme.standalone, 24
Fetch your documentation here.
path (acme.challenges.HTTP01 property), 6
phones (acme.messages.Registration property), 21
poll() (acme.client.Client method), 11
poll() (acme.client.ClientV2 method), 13
poll_and_finalize() (acme.client.ClientV2 method), 13
poll_and_request_issuance() (acme.client.Client method), 11
poll_authorizations() (acme.client.ClientV2 method), 14
PollError, 17
PORT (acme.challenges.HTTP01Response attribute), 5
PORT (acme.challenges.TLSALPN01Response attribute), 6
post() (acme.client.ClientNetwork method), 16
probe_cert() (acme.challenges.TLSALPN01Response method), 6
query_registration() (acme.client.Client method), 10
query_registration() (acme.client.ClientV2 method), 13
refresh() (acme.client.Client method), 12
register() (acme.client.Client method), 10
register() (acme.messages.Directory class method), 20
Registration (class in acme.messages), 20
RegistrationResource (class in acme.messages), 21
REPLAY_NONCE_HEADER (acme.client.ClientNetwork attribute), 16
request_challenges() (acme.client.Client method), 10
request_domain_challenges() (acme.client.Client method), 11
request_issuance() (acme.client.Client method), 11
resolved_combinations (acme.messages.Authorization property), 22
Resource (class in acme.fields), 18
Resource (class in acme.messages), 20
resource() (in module acme.fields), 18
ResourceBody (class in acme.messages), 20
ResourceWithURI (class in acme.messages), 20
response (acme.standalone.HTTP01RequestHandler.HTTP01Resource property), 24
response() (acme.challenges.KeyAuthorizationChallenge method), 4
response_and_validation() (acme.challenges.KeyAuthorizationChallenge method), 4
response_cls (acme.challenges.HTTP01 attribute), 6
response_cls (acme.challenges.TLSALPN01 attribute), 7
retry_after() (acme.client.ClientBase class method), 10
Revocation (class in acme.messages), 23
revoke() (acme.client.BackwardsCompatibleClientV2 method), 15
revoke() (acme.client.Client method), 12
revoke() (acme.client.ClientV2 method), 14
rfc3339() (in module acme.fields), 18
RFC3339Field (class in acme.fields), 18
SchemaValidationError, 16
serve_forever() (acme.standalone.BaseDualNetworkedServers method), 24
server_bind() (acme.standalone.TLSServer method), 24
shutdown_and_server_close() (acme.standalone.BaseDualNetworkedServers method), 24
simple_verify() (acme.challenges.DNS01Response method), 4
simple_verify() (acme.challenges.HTTP01Response method), 5
simple_verify() (acme.challenges.TLSALPN01Response method), 7
Status (class in acme.messages), 19
terms_of_service (acme.messages.Directory.Meta property), 20
timeout (acme.errors.PollError property), 17
timeout (acme.standalone.HTTP01RequestHandler property), 25
TimeoutError, 17
TLSALPN01 (class in acme.challenges), 7
TLSALPN01Response (class in acme.challenges), 6
TLSALPN01Server (class in acme.standalone), 24
TLSServer (class in acme.standalone), 24
to_partial_json() (acme.challenges.KeyAuthorizationChallengeResponse method), 4
to_partial_json() (acme.challenges.KeyAuthorizationChallengeResponse method), 4
to_partial_json() (acme.challenges.UnrecognizedChallenge method), 3
to_partial_json() (acme.messages.ChallengeBody method), 4
to_partial_json() (acme.messages.ChallengeBody method), 4
to_partial_json() (acme.messages.Directory method), 20
to_partial_json() (acme.messages.Directory method), 20
to_partial_json() (acme.messages.Registration method), 21
UnexpectedUpdate, 16
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
ValidationError, 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