ACME Working Group
Internet-Draft
Intended status: Standards Track
Expires: 31 December 2022

O. Friel
R. Barnes
Cisco
T. Hollebeek
DigiCert
M. Richardson
Sandelman Software Works
29 June 2022

ACME for Subdomains
draft-ietf-acme-subdomains-04

Abstract

This document outlines how ACME can be used by a client to obtain a certificate for a subdomain identifier from a certification authority. The client has fulfilled a challenge against a parent domain but does not need to fulfill a challenge against the explicit subdomain as certification authority policy allows issuance of the subdomain certificate without explicit subdomain ownership proof.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 31 December 2022.

Table of Contents

1. Introduction

ACME [RFC8555] defines a protocol that a certification authority (CA) and an applicant can use to automate the process of domain name ownership validation and X.509v3 (PKIX) [RFC5280] certificate issuance. This document outlines how ACME can be used to issue subdomain certificates, without requiring the ACME client to explicitly fulfill an ownership challenge against the subdomain identifiers - the ACME client need only fulfill an ownership challenge against a parent domain identifier.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are defined in DNS Terminology [RFC8499] and are reproduced here:

* Label: An ordered list of zero or more octets that makes up a portion of a domain name. Using graph theory, a label identifies one node in a portion of the graph of all possible domain names.

* Domain Name: An ordered list of one or more labels.

* Subdomain: "A domain is a subdomain of another domain if it is contained within that domain. This relationship can be tested by seeing if the subdomain's name ends with the containing domain's name." (Quoted from [RFC1034], Section 3.1) For example, in the host name "nnn.mmm.example.com", both "mmm.example.com" and "nnn.mmm.example.com" are subdomains of "example.com". Note that the comparisons here are done on whole labels; that is, "ooo.example.com" is not a subdomain of "oo.example.com".

* Fully-Qualified Domain Name (FQDN): This is often just a clear way of saying the same thing as "domain name of a node", as outlined above. However, the term is ambiguous. Strictly speaking, a fully-qualified domain name would include every label, including the zero-length label of the root: such a name would be written "www.example.net." (note the terminating dot). But, because every name eventually shares the common root, names are often written relative to the root (such as "www.example.net") and are still called "fully qualified". This term first appeared in [RFC0819]. In this document, names are often written relative to the root.

The following additional terms are used in this document:

* Certification Authority (CA): An organization that is responsible for the creation, issuance, revocation, and management of Certificates. The term applies equally to both Root CAs and Subordinate CAs. Refer to [RFC5280] for detailed information on Certification Authorities.

* CSR: Certificate Signing Request as defined in [RFC2986]

* Parent Domain: a domain is a parent domain of a subdomain if it contains that subdomain, as per the [RFC8499] definition of subdomain. For example, for the host name "nnn.mmm.example.com", both "mmm.example.com" and "example.com" are parent domains of "nnn.mmm.example.com". Note that the comparisons here are done on whole labels; that is, "oo.example.com" is not a parent domain of "ooo.example.com"

3. ACME Workflow and Identifier Requirements

A typical ACME workflow for issuance of certificates is as follows:

1. client POSTs a newOrder request that contains a set of "identifiers"

2. server replies with a set of "authorizations" and a "finalize" URI

3. client sends POST-as-GET requests to retrieve the "authorizations", with the downloaded "authorization" object(s) containing the "identifier" that the client must prove that they control, and a set of associated "challenges", one of which the client must fulfill

4. client proves control over the "identifier" in the "authorization" object by completing one of the specified challenges, for example, by publishing a DNS TXT record

5. client POSTs a CSR to the "finalize" API

6. server replies with an updated order object that includes a "certificate" URI

7. client sends POST-as-GET request to the "certificate" URI to download the certificate

ACME places the following restrictions on "identifiers":

* [RFC8555] section 7.1.3: The authorizations required are dictated by server policy; there may not be a 1:1 relationship between the order identifiers and the authorizations required.

* [RFC8555] section 7.1.4: the only type of "identifier" defined by the ACME specification is an FQDN: "The only type of identifier defined by this specification is a fully qualified domain name (type: "dns"). The domain name MUST be encoded in the form in which it would appear in a certificate."

* [RFC8555] section 7.4: the "identifier" in the CSR request must match the "identifier" in the newOrder request: "The CSR MUST indicate the exact same set of requested identifiers as the initial newOrder request."

* [RFC8555] section 8.3: the "identifier", or FQDN, in the "authorization" object must be used when fulfilling challenges via HTTP: "Construct a URL by populating the URL template ... where the domain field is set to the domain name being verified"

* [RFC8555] section 8.4: the "identifier", or FQDN, in the "authorization" object must be used when fulfilling challenges via DNS: "The client constructs the validation domain name by prepending the label "_acme-challenge" to the domain name being validated."

ACME does not mandate that the "identifier" in a newOrder request matches the "identifier" in "authorization" objects.

4. ACME Issuance of Subdomain Certificates

As noted in the previous section, ACME does not mandate that the "identifier" in a newOrder request matches the "identifier" in "authorization" objects. This means that the ACME specification does not preclude an ACME server processing newOrder requests and issuing certificates for a subdomain without requiring a challenge to be fulfilled against that explicit subdomain.

ACME server policy could allow issuance of certificates for a subdomain to a client where the client only has to fulfill an authorization challenge for a parent domain of that subdomain. This allows a flow where a client proves ownership of, for example, "example.org" and then successfully obtains a certificate for "sub.example.org".

ACME server policy is out of scope of this document, however, some commentary is provided in Section 7.2.

Clients need a mechanism to instruct the ACME server that they are requesting authorization for all subdomains subordinate to the specified domain, as opposed to just requesting authorization for an explicit domain identifier. Clients need a mechanism to do this in both newAuthz and newOrder requests. ACME servers need a mechanism to indicate to clients that authorization objects are valid for all subdomains under the specified domain. These are described in this section.

4.1. Authorization Object

ACME [RFC8555] section 7.1.4 defines the authorization object. When ACME server policy allows authorization for subdomains subordinate to a domain, the server indicates this by including the "subdomainAuthAllowed" flag in the authorization object for that domain identifier:

subdomainAuthAllowed (optional, boolean): This field MUST be present and true for authorizations where ACME server policy allows certificates to be issued for any subdomain subordinate to the domain specified in the 'identifier' field of the authorization object.

The following example shows an authorization object for the domain example.org where the authorization covers the subdomains subordinate to example.org.

```
{
  "status": "valid",
  "expires": "2015-03-01T14:09:07.99Z",

  "identifier": {
    "type": "dns",
    "value": "example.org"
  },

  "challenges": [
    {
      "url": "https://example.com/acme/chall/prV_B7yEyA4",
      "type": "http-01",
      "status": "valid",
      "token": "DGyRejmCefe7v4NfDGDKfA",
      "validated": "2014-12-01T12:05:58.16Z"
    }
  ],

  "subdomainAuthAllowed": true
}
```

If the "subdomainAuthAllowed" field is not included, then the assumed default value is false.

4.2. Pre-Authorization

The standard ACME workflow has authorization objects created reactively in response to a certificate order. ACME also allows for pre-authorization, where clients obtain authorization for an identifier proactively, outside of the context of a specific issuance. With the ACME pre-authorization flow, a client can pre-authorize for a domain once, and then issue multiple newOrder requests for certificates with identifiers in the subdomains subordinate to that domain.

ACME [RFC8555] section 7.4.1 defines the "identifier" object for newAuthz requests. One additional field for the "identifier" object is defined:

subdomainAuthAllowed (optional, boolean): An ACME client sets this flag to indicate to the server that it is requesting an authorization for the subdomains subordinate to the specified domain identifier value

Clients include the flag in the "identifier" object of newAuthz requests to indicate that they are requesting a subdomain authorization. In the following example newAuthz payload, the client is requesting pre-authorization for the subdomains subordinate to example.org.

```
"payload": base64url({
  "identifier": {
    "type": "dns",
    "value": "example.org",
    "subdomainAuthAllowed": true
  }
})
```

If the server is willing to allow a single authorization for the subdomains, and there is not an existing authorization object for the identifier, then it will create an authorization object and include the "subdomainAuthAllowed" flag with value of true. If the server policy does not allow creation of subdomain authorizations subordinate to that domain, the server can create an authorization object for the indicated identifier, and include the "subdomainAuthAllowed" flag with value of false. In both scenarios, handling of the pre-authorization follows the process documented in ACME section 7.4.1.

4.3. New Orders

Clients need a mechanism to optionally indicate to servers whether or not they are authorized to fulfill challenges against parent domains for a given identifier FQDN. For example, if a client places an order for an identifier foo.bar.example.org, and is authorized to fulfill a challenge against the parent domains bar.example.org or example.org, then the client needs a mechanism to indicate control over the parent domains to the ACME server.

This can be achieved by adding an optional field "parentDomain" to the "identifiers" field in the order object:

parentDomain (optional, string): This is a parent domain of the requested identifier. The client MUST be able to fulfill a challenge against the parent domain.

This field specifies a parent domain of the identifier that the client has DNS control over, and is capable of fulfilling challenges against. Based on server policy, the server can choose to issue a challenge against any parent domain of the identifier up to and including the specified "parentDomain", and create a corresponding authorization object against the chosen identifier.

In the following example newOrder payload, the client requests a certificate for identifier foo.bar.example.org and indicates that it can fulfill a challenge against the parent domain bar.example.org. The server can then choose to issue a challenge against either foo.bar.example.org or bar.example.org identifiers.

```
"payload": base64url({
       "identifiers": [
         { "type": "dns",
           "value": "foo.bar.example.org",
           "parentDomain": "bar.example.org"  }
       ],
       "notBefore": "2016-01-01T00:04:00+04:00",
       "notAfter": "2016-01-08T00:04:00+04:00"
     })
```

In the following example newOrder payload, the client requests a certificate for identifier foo.bar.example.org and indicates that it can fulfill a challenge against the parent domain example.org. The server can then choose to issue a challenge against any one of foo.bar.example.org, bar.example.org or example.org identifiers.

```
"payload": base64url({
       "identifiers": [
         { "type": "dns",
           "value": "foo.bar.example.org",
           "parentDomain": "example.org"  }
       ],
       "notBefore": "2016-01-01T00:04:00+04:00",
       "notAfter": "2016-01-08T00:04:00+04:00"
     })
```

If the client is unable to fulfill authorizations against parent domain, the client should not include the "parentDomain" field.

Server newOrder handling generally follows the process documented ACME section 7.4. If the server is willing to allow subdomain authorizations for the domain specified in "parentDomain", then it creates an authorization object against that parent domain and includes the "subdomainAuthAllowed" flag with a value of true. If the server policy does not allow creation of subdomain authorizations against that parent domain, then it can create an authorization object for the indicated identifier value, and includes the "subdomainAuthAllowed" flag with value of false.

4.4. Directory Object Metadata

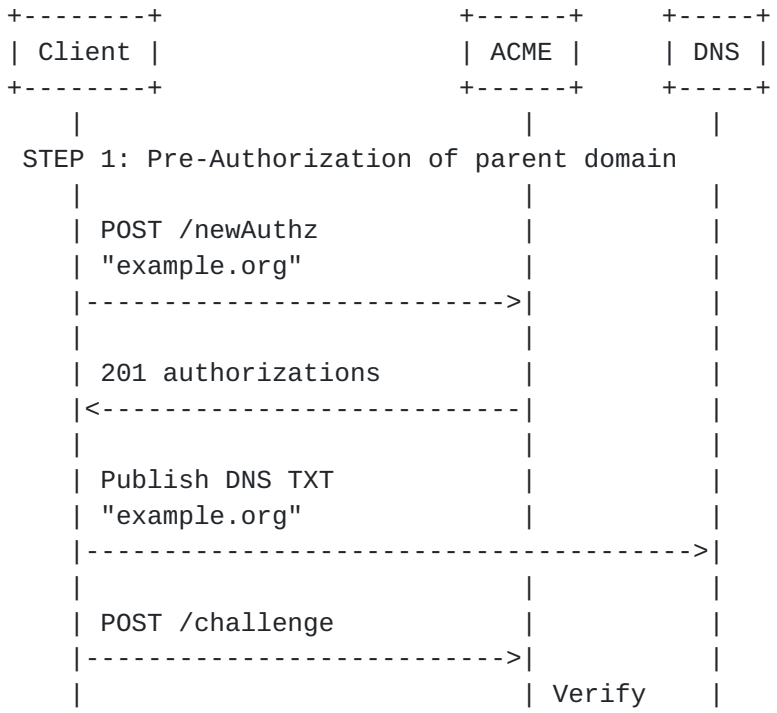
An ACME server can advertise support for authorization of subdomains by including the following boolean flag in its "ACME Directory Metadata Fields" registry:

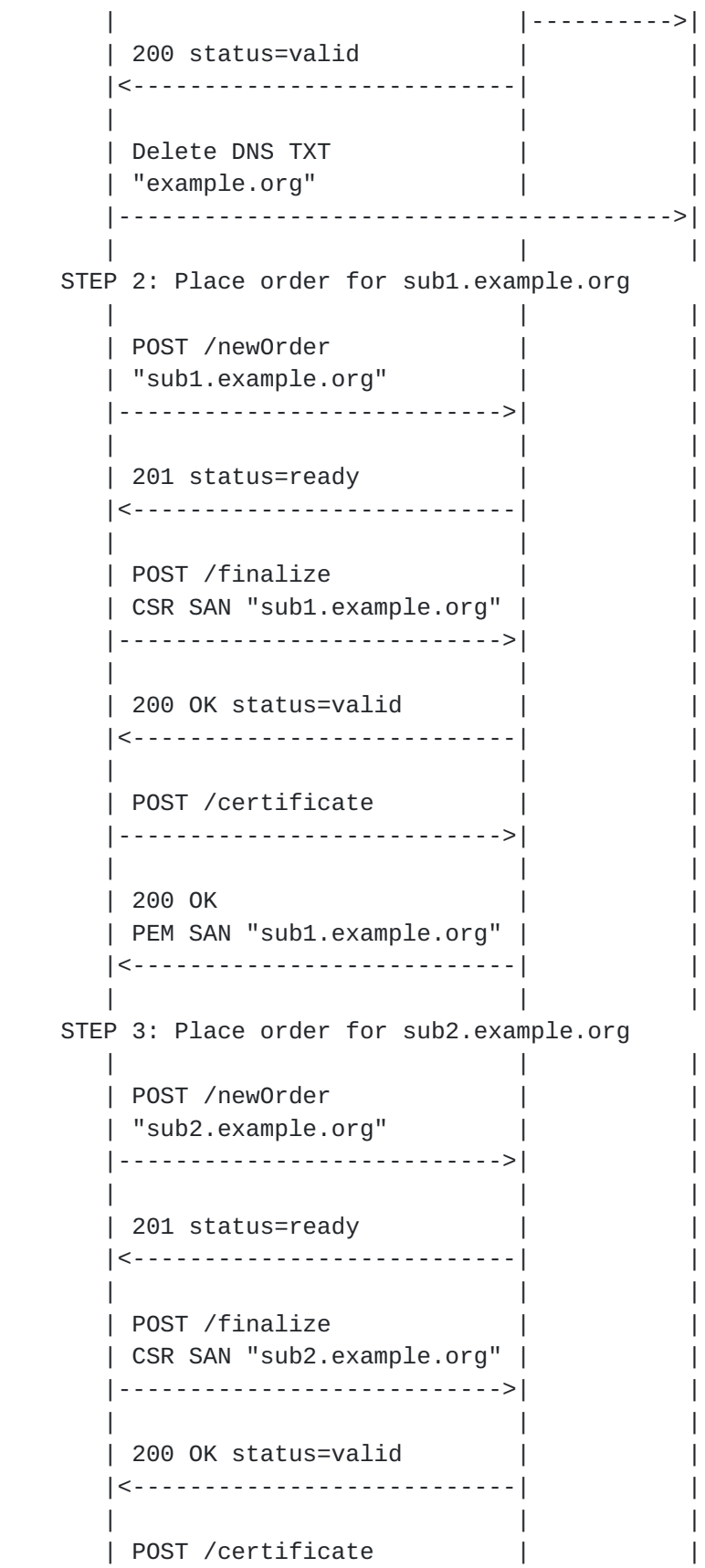
subdomainAuthAllowed (optional, bool): Indicates if an ACME server supports authorization of subdomains.

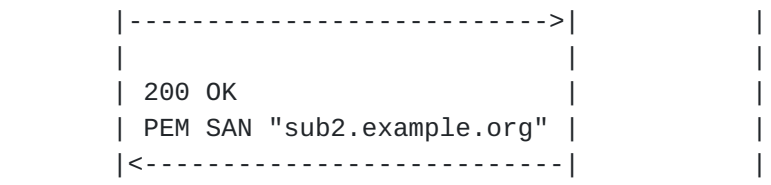
If not specified, then no default value is assumed. If an ACME server supports authorization of subdomains, it can indicate this by including this field with a value of "true".

5. Illustrative Call Flow

The call flow illustrated here uses the ACME pre-authorization flow using DNS-based proof of ownership.




200 status=valid
Delete DNS TXT
"example.org"
STEP 2: Place order for sub1.example.org
POST /newOrder
"sub1.example.org"
201 status=ready
POST /finalize
CSR SAN "sub1.example.org"
200 OK status=valid
POST /certificate
200 OK
PEM SAN "sub1.example.org"
STEP 3: Place order for sub2.example.org
POST /newOrder
"sub2.example.org"
201 status=ready
POST /finalize
CSR SAN "sub2.example.org"
200 OK status=valid
POST /certificate



* STEP 1: Pre-authorization of parent domain

  The client sends a newAuthz request for the parent domain including the "subdomainAuthAllowed" flag in the identifier object.

```
POST /acme/new-authz HTTP/1.1
Host: example.com
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://example.com/acme/acct/evOfKhNU60wg",
    "nonce": "uQpSjlRb4vQVCjVYAyyUWg",
    "url": "https://example.com/acme/new-authz"
  }),
  "payload": base64url({
    "identifier": {
      "type": "dns",
      "value": "example.org",
      "subdomainAuthAllowed": true
    }
  }),
  "signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"
}
```

The server creates and returns an authorization object for the identifier including the "subdomainAuthAllowed" flag. The object is initially in "pending" state.

```
{
  "status": "pending",
  "expires": "2015-03-01T14:09:07.99Z",

  "identifier": {
    "type": "dns",
    "value": "example.org"
  },

  "challenges": [
    {
      "url": "https://example.com/acme/chall/prV_B7yEyA4",
      "type": "http-01",
      "status": "pending",
      "token": "DGyRejmCefe7v4NfDGDKfA",
      "validated": "2014-12-01T12:05:58.16Z"
    }
  ],

  "subdomainAuthAllowed": true
}
```

Once the client completes the challenge, the server will transition the authorization object and associated challenge object status to "valid". The call flow above illustrates the ACME server replying to the client's challenge with status of "valid" after the ACME server has validated the DNS challenge.

However, the validation flow may take some time. If this is the case, the ACME server may reply to the client's challenge immediately with a status of "processing", and the client will then need to poll the authorization resource to see when it is finalized. Refer to ACME [RFC8555] section 7.5.1 for more details.

* STEP 2: The client places a newOrder for sub1.example.org

  The client sends a newOrder request to the server and includes the subdomain identifier. Note that the identifier is a subdomain of the parent domain that has been pre-authorised in step 1. The client does not need to include the "subdomainAuthAllowed" field in the "identifier" object as it has already pre-authorized the parent domain.

```
POST /acme/new-order HTTP/1.1
Host: example.com
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://example.com/acme/acct/evOfKhNU60wg",
    "nonce": "5XJ1L3lEkMG7tR6pA00clA",
    "url": "https://example.com/acme/new-order"
  }),
  "payload": base64url({
    "identifiers": [
      { "type": "dns", "value": "sub1.example.org" }
    ],
    "notBefore": "2016-01-01T00:04:00+04:00",
    "notAfter": "2016-01-08T00:04:00+04:00"
  }),
  "signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g"
}
```

As an authorization object already exists for the parent domain, the server replies with an order object with a status of "ready" that includes a link to the existing "valid" authorization object.

```
HTTP/1.1 201 Created
Replay-Nonce: MYAuvOpaoIiywTezizk5vw
Link: <https://example.com/acme/directory>;rel="index"
Location: https://example.com/acme/order/TOlocE8rfgo

{
  "status": "ready",
  "expires": "2016-01-05T14:09:07.99Z",

  "notBefore": "2016-01-01T00:00:00Z",
  "notAfter": "2016-01-08T00:00:00Z",

  "identifiers": [
    { "type": "dns", "value": "sub1.example.org" }
  ],

  "authorizations": [
    "https://example.com/acme/authz/PAniVnsZcis"
  ],

  "finalize": "https://example.com/acme/order/TOlocrfgo/finalize"
}
```

The client can proceed to finalize the order and download the certificate for sub1.example.org.

* STEP 3: The client places a newOrder for sub2.example.org

  The client sends a newOrder request to the server and includes the subdomain identifier. Note that the identifier is a subdomain of the parent domain that has been pre-authorised in step 1. The client does not need to include the "subdomainAuthAllowed" field in the "identifier" object as it has already pre-authorized the parent domain.

```
POST /acme/new-order HTTP/1.1
Host: example.com
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://example.com/acme/acct/evOfKhNU60wg",
    "nonce": "5XJ1L3lEkMG7tR6pA00clA",
    "url": "https://example.com/acme/new-order"
  }),
  "payload": base64url({
    "identifiers": [
      { "type": "dns", "value": "sub2.example.org" }
    ],
    "notBefore": "2016-01-01T00:04:00+04:00",
    "notAfter": "2016-01-08T00:04:00+04:00"
  }),
  "signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g"
}
```

As an authorization object already exists for the parent domain, the server replies with an order object with a status of "ready" that includes a link to the existing "valid" authorization object.

```
HTTP/1.1 201 Created
Replay-Nonce: MYAuvOpaoIiywTezizk5vw
Link: <https://example.com/acme/directory>;rel="index"
Location: https://example.com/acme/order/TOlocE8rfgo

{
  "status": "ready",
  "expires": "2016-01-05T14:09:07.99Z",

  "notBefore": "2016-01-01T00:00:00Z",
  "notAfter": "2016-01-08T00:00:00Z",

  "identifiers": [
    { "type": "dns", "value": "sub2.example.org" }
  ],

  "authorizations": [
    "https://example.com/acme/authz/PAniVnsZcis"
  ],

  "finalize": "https://example.com/acme/order/ROni7rdde/finalize"
}
```

The client can proceed to finalize the order and download the certificate for sub2.example.org.

6. IANA Considerations

6.1. Authorization Object Fields Registry

The following field is added to the "ACME Authorization Object Fields" registry defined in ACME [RFC8555].

Field Name	Field Type	Configurable	Reference
subdomainAuthAllowed	boolean	false	RFC XXXX

6.2. Directory Object Metadata Fields Registry

The following field is added to the "ACME Directory Metadata Fields" registry defined in ACME [RFC8555].

Field Name	Field Type	Reference
subdomainAuthAllowed	boolean	RFC XXXX

7. Security Considerations

This document documents enhancements to ACME [RFC8555] that optimize the protocol flows for issuance of certificates for subdomains. The underlying goal of ACME for Subdomains remains the same as that of ACME: managing certificates that attest to identifier/key bindings for these subdomains. Thus, ACME for Subdomains has the same two security goals as ACME:

1. Only an entity that controls an identifier can get an authorization for that identifier

2. Once authorized, an account key's authorizations cannot be improperly used by another account

ACME for Subdomains makes no changes to:

* account or account key management

* ACME channel establishment, security mechanisms or threat model

* Validation channel establishment, security mechanisms or threat model

Therefore, all Security Considerations in ACME in the following areas are equally applicable to ACME for Subdomains:

* Threat Model

* Integrity of Authorizations

* Denial-of-Service Considerations

* Server-Side Request Forgery

* CA Policy Considerations

Some additional comments on ACME server policy are given in the following section.

7.1. Subdomain Determination

The [RFC8499] definition of a subdomain is reproduced in Section 2. When comparing domains to determine if one is a subdomain of the other, it is important to compare entire labels, and not rely on a string prefix match. Relying on string prefix matches may yield incorrect results.

7.2. ACME Server Policy Considerations

The ACME for Subdomains and the ACME specifications do not mandate any specific ACME server or CA policies, or any specific use cases for issuance of certificates. For example, an ACME server could be used:

* to issue Web PKI certificates where the ACME server must comply with CA/Browser Forum [CAB] Baseline Requirements.

* as a Private CA for issuance of certificates within an organization. The organization could enforce whatever policies they desire on the ACME server.

* for issuance of IoT device certificates. There are currently no IoT device certificate policies that are generally enforced across the industry. Organizations issuing IoT device certificates can enforce whatever policies they desire on the ACME server.

ACME server policy could specify whether:

* issuance of subdomain certificates is allowed based on proof of ownership of a parent domain

* issuance of subdomain certificates is allowed, but only for a specific set of parent domains

* whether DNS based proof of ownership, or HTTP based proof of ownership, or both, are allowed

ACME server policy specification is explicitly out of scope of this document.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[CAB] CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates", n.d., <https://cabforum.org/wp-content/uploads/CA-Browser-Forum-BR-1.7.1.pdf>.

[RFC0819] Su, Z. and J. Postel, "The Domain Naming Convention for Internet User Applications", RFC 819, DOI 10.17487/RFC0819, August 1982, <https://www.rfc-editor.org/info/rfc819>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <https://www.rfc-editor.org/info/rfc2986>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <https://www.rfc-editor.org/info/rfc8555>.

Authors' Addresses

Owen Friel
Cisco
Email: ofriel@cisco.com

Richard Barnes
Cisco
Email: rlb@ipv.sx

Tim Hollebeek
DigiCert
Email: tim.hollebeek@digicert.com

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca