Automated Certificate Management Environment
Internet-Draft
Intended status: Standards Track
Expires: 23 April 2026

G. De Marco
independent
B. Pitman

20 October 2025

Automatic Certificate Management Environment (ACME) with OpenID Federation 1.0
draft-demarco-acme-openid-federation-01

Abstract

The Automatic Certificate Management Environment (ACME) protocol allows server operators to obtain TLS certificates for their websites, based on a demonstration of control over the website's domain via a fully-automated challenge/response protocol.

OpenID Federation 1.0 defines how to build a trust infrastructure using a trusted third-party model. It uses a trust evaluation mechanism to attest to the possession of private keys, protocol specific metadata and miscellaneous administrative and technical information related to a specific entity.

This document defines how X.509 certificates associated with a given OpenID Federation Entity can be issued by an X.509 Certification Authority through the ACME protocol to the organizations which are part of a federation built on top of OpenID Federation 1.0.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-demarco-acme-openid-federation/.

Discussion of this document takes place on the Automated Certificate Management Environment Working Group mailing list (mailto:acme@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/acme/. Subscribe at https://www.ietf.org/mailman/listinfo/acme/.

Source for this draft and an issue tracker can be found at https://github.com/peppelinux/draft-demarco-acme-openid-federation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document describes extensions to the ACME protocol that integrate with OpenID Federation 1.0, allowing an ACME server to issue X.509 Certificates associated with a given OpenID Federation entity. An entity can be a school, government agency, corporation, automated system, or any participant that wishes to join the Federation.

In a multilateral federation, composed of thousands of entities belonging to different organizations, all the participants adhere to the same regulation or trust framework. OpenID Federation 1.0 allows each participant to recognize other participants using a trust evaluation mechanism, with RESTful services and cryptographic materials.

Federation members declare what kind Entities they are using a basic OpenID Federation component called an Entity Configuration, a signed JSON Web Token published in a well-known resource. This document defines new OpenID Federation Entity Types for certificate Requestors and Issuers, facilitating automated discovery of an issuer's ACME API.

X.509 Certificates can be provided to one or more such entities, without having pre-established any direct relationship or contract.

These Certificates are useful for integrating with existing systems and protocols which require X.509 Certificates. For example, RADIUS ([RFC2865]) deployments often require Certificates to authenticate clients, or these Certificates could also be used for TLS [RFC8446]. In order to accommodate a variety of use cases, this document adds no requirement that CAs conform to any particular issuance profile, because fields of the X.509 Certificates like the Common Names, Subject Alternative Names or Key Usages may depend on details of those existing systems.

This document extends [RFC5280], ACME ([RFC8555]) and OpenID Federation 1.0 ([OPENID-FED]) in the following ways:

* It defines a new ACME Identifier type called openid-federation (Section 4).

* It defines a new ACME challenge type called openid-federation-01 (Section 5).

* It defines new OpenID Federation 1.0 Entity Types acme_issuer (Section 7.1) and acme_requestor (Section 6).

* It defines how OpenID Federation 1.0 Superior Entities can use Subordinate Statements to publish X.509 Certificates previously issued with ACME (Section 8).

* It defines a new SubjectAlternativeName type id-on-OpenIdFederationEntityId and a corresponding OID so that OpenID Federation 1.0 Entity Identifiers can be included in X.509 Certificates if an Issuer wishes (Section 9).

2. Terminology

The terms "Federation Entity", "Trust Anchor", "Entity Configuration", "Subordinate Statement", "Superior Entity", "Immediate Superior Entity", "Federation Entity Keys", "Federation Entity Discovery", "Trust Mark", "Trust Chain", and "Resolved Metadata" used in this document are defined in Section 1.2 of [OPENID-FED].

The term "Certificate Signing Request" (CSR) used in this document is defined as a "Certification Request" in [RFC2986]. The term "Certification Authority" used in this document is defined in [RFC5280]. The terms "ACME Client" and "ACME Server" are defined in [RFC8555].

The specification also defines the following terms:

ACME Identifier: An identifier in the sense of [RFC8555]. Some identifier that the ACME client proves control of to the ACME server.

Entity Identifier: A URL that uniquely identifies an Entity in OpenID Federation, as defined in Section 1.2 of [OPENID-FED].

Requestor: A Federation Entity which wants to request X.509

Certificates. It operates a web server for hosting its Entity Configuration. It also operates an ACME client, extended according to this document.

Certificate Issuer (or Issuer): A Federation Entity which issues X.509 Certificates. It operates a web server for hosting its Entity Configuration. It also operates an ACME server, extended according to this document.

3. Conventions and Definitions

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.

4. OpenID Federation ACME Identifier

This document defines a new ACME Identifier type for OpenID Federation Entities, openid-federation, whose value is the sub parameter of the Requestor’s Entity Configuration, as defined in Section 1.2 of [OPENID-FED].

For example, the ACME Identifier corresponding to the example Entity Configuration in Section 6 is:

```
{"type": "openid-federation", "value": "https://requestor.example.com"}
```

The openid-federation ACME Identifier type MUST NOT be validated except by the openid-federation-01 challenge.

5. OpenID Federation Challenge Type

The OpenID Federation challenge type allows a Requestor to prove control of a Federation Entity using the trust evaluation mechanism provided by [OPENID-FED]. The Requestor demonstrates control of a cryptographic public key published in its OpenID Federation Entity Configuration.

There are two ways the Certificate Issuer is able to check if a Requestor is part of the federation:

* The Requestor provides a Trust Chain when solving the ACME challenge. This is RECOMMENDED since it reduces the effort of the Certificate Issuer in evaluating the trust to the Requestor.

* The Requestor doesn’t provide a Trust Chain in the challenge solution.

The openid-federation-01 ACME challenge object has the following format:

type (required, string): The string "openid-federation-01"

token (required, string): A random value that uniquely identifies the challenge. This value MUST have at least 128 bits of entropy. It MUST NOT contain any characters outside the base64url alphabet as described in Section 5 of [RFC4648]. Trailing ’=’ padding characters MUST be stripped. See [RFC4086] for additional information on randomness requirements.

trustAnchors (optional, array of string): An array of strings containing the Entity Identifiers of the Issuer’s Trust Anchors. When solving the challenge, the Requestor can construct a Trust Chain from itself to one of these Trust Anchors. It is RECOMMENDED that the Issuer includes this field to make it easier for the Requestor to construct a Trust Chain.

A non-normative example of a challenge with trustAnchors specified:

```
{
  "type": "openid-federation-01",
  "url": "https://issuer.example.com/acme/chall/prV_B7yEyA4",
  "status": "pending",
  "token": "LoqXcYV8q5ONbJQxbmR7SCTNo3tiAXDfowyjxAjEuX0",
  "trustAnchors": [
    "https://trust-anchor-1.example.com",
    "https://trust-anchor-2.example.com"
  ]
}
```

The openid-federation-01 challenge MUST NOT be used to validate possession of any ACME Identifiers except openid-federation ACME Identifiers.

The Requestor responds to the challenge with an object with the following format:

sig (required, string): the compact JSON serialization (as described in Section 7.1 of [RFC7515]) of a JWS, signing the key authorization encoded in UTF-8. The key authorization is computed from the token in the challenge and the Requestor’s ACME account key, as defined in Section 8.1 of [RFC8555]. The signature must be made by one of the keys published in the Requestor’s acme_requestor metadata in its

Entity Configuration, as specified in Section 6. The JWS MUST include a kid header parameter corresponding to the key used to sign the key authorization and a typ header parameter set to "signed-acme-challenge+jwt".

trustChain (optional, array of string): an array of strings containing signed JWTs, representing a Trust Chain from the Requestor to one of the Issuer's Trust Anchors (see Section 4 of [OPENID-FED]). The Resolved Metadata of the Trust Chain subject MUST contain acme_requestor metadata that contains the key used to compute sig. It is RECOMMENDED that the Requestor includes this field. If the Requestor cannot construct a Trust Chain to one of the Trust Anchors indicated by the Issuer, or if no Trust Anchors were indicated, it MAY use some other Trust Anchor that it believes the Issuer trusts. If the Requestor cannot construct a Trust Chain to any Trust Anchor, it MAY omit the trustChain field from the challenge response.

A non-normative example for an authorization with trustChain specified:

```
POST /acme/chall/prV_B7yEyA4
Host: issuer.example.com
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://issuer.example.com/acme/acct/evOfKhNU60wg",
    "nonce": "UQI1PoRi5OuXzxuX7V7wL0",
    "url": "https://issuer.example.com/acme/chall/prV_B7yEyA4"
  }),
  "payload": base64url({
   "sig": "wQAvHlPV1tVxRW0vZUa4BQ...",
   "trustChain": ["eyJhbGciOiJFU...", "eyJhbGci..."]
  }),
  "signature": "Q1bURgJoEslbD1c5...3pYdSMLio57mQNN4"
}
```

On receiving a challenge response, the Certificate Issuer verifies that the Requestor is trusted. If the Requestor did not provide a trustChain, the Issuer MUST perform Federation Entity Discovery (Section 10 of [OPENID-FED]) to obtain a Trust Chain for the Requestor.

Once it has obtained a Trust Chain, the Issuer evaluates the entity's Resolved Metadata, and verifies:

* That the requested openid-federation ACME Identifier value matches the sub parameter of the Requestor's Entity Configuration.

* That there is a key in the acme_requestor metadata (Section 6) of the Requestor's Resolved Metadata with a kid matching the kid claim in the challenge response.

* That the sig field of the payload is the compact JSON serialization of a valid JWS signing the key authorization, using the above public key.

If all of the above checks succeed, then the validation is successful. Otherwise, it has failed. In either case, the Certificate Issuer responds according to Section 7.5.1 of [RFC8555]. If the Issuer fails to verify OpenID Federation trust, the problem document SHOULD contain a subproblem of type urn:ietf:params:acme:error:openIDFederationEntity constructed as discussed in Section 11.

A non-normative example for the challenge object post-validation:

```
{
  "type": "openid-federation-01",
  "url": "https://issuer.example.com/acme/chall/prV_B7yEyA4",
  "status": "valid",
  "validated": "2024-10-01T12:05:13.72Z",
  "token": "LoqXcYV8q5ONbJQxbmR7SCTNo3tiAXDfowyjxAjEuX0"
}
```

6. Requestor Entity Configuration Metadata

The Requestor MUST publish in its Entity Configuration an acme_requestor metadata containing a JWK set, according to Section 5.2.1 of [OPENID-FED]. The keys in the set are used to respond to ACME challenges.

The following is a non-normative example of an Entity Configuration including the acme_requestor metadata and using the jwks metadata parameter.

```
{
  "iss": "https://requestor.example.com",
  "sub": "https://requestor.example.com",
  "iat": 1516239022,
  "exp": 1516298022,
  "jwks": {
    "keys": [
      {
        "kty": "RSA",
        "alg": "RS256",
        "use": "sig",
        "kid": "NzbLsXh8uDCcd-6MNwXF4W_7noWXFZAfHkxZsRGC9Xs",
        "n": "pnXBOusEANuug6ewezb9J_...",
        "e": "AQAB"
      }
    ]
  },
  "metadata": {
    "acme_requestor": {
      "jwks": {
        "keys": [
          {
            "kty": "RSA",
            "kid": "SUdtUndEWVY2cUFDeD...",
            "n": "y_Zc8rByfeRIC9fFZrD...",
            "e": "AQAB"
          },
          {
            "kty": "EC",
            "kid": "MFYycG1raTI4SkZvVDBIMF9CNGw3VEZYUmxQLVN2T21nSWlkd3",
            "crv": "P-256",
            "x": "qAOdPQROkHfZY1daGofOmSNQWpYK8c9G2m2Rbkpbd4c",
            "y": "G_7fF-T8n2vONKM15Mzj4KR_shvHBxKGjMosF6FdoPY"
          }
        ]
      }
    }
  }
}
```


The Issuer MUST only use the Requestor’s acme_requestor to validate an ACME challenge. Therefore, after completing the challenge, the Requestor MAY remove the acme_requestor metadata from its Entity Configuration.

7. Issuer Discovery

The Requestor's ACME client may either be configured to use a particular ACME server, or to automatically discover a Certificate Issuer through the OpenID Federation.

In order to be discoverable, the Issuer MUST publish the entity type acme_issuer in its Entity Configuration, according to Section 7.1.

[OPENID-FED] describes a variety of patterns and generic mechanisms for discovering Federation Entities. This section describes how Issuers may make themselves discoverable in a Federation by Requestors.

TODO: include a specific section reference once https://github.com/openid/federation/pull/265 lands in OPENID-FED

7.1. Issuer Metadata

The Issuer MUST publish its Entity Configuration including the acme_issuer Entity Type metadata within it. The acme_issuer metadata contains one parameter, directory_url, which is the URL of the ACME Directory, as defined in Section 7.1.1 of [RFC8555].

Requestors MUST use the ACME Directory provided in the Issuer's Entity Configuration for client configuration of ACME endpoints.

The following is a non-normative example of an Entity Configuration including the acme_issuer metadata:

```
{
  "iss": "https://issuer.example.com",
  "sub": "https://issuer.example.com",
  "iat": 1516239022,
  "exp": 1516298022,
  "jwks": {
    "keys": [
      {
        "kty": "RSA",
        "alg": "RS256",
        "use": "sig",
        "kid": "NzbLsXh8uDCcd-6MNwXF4W_7noWXFZAfHkxZsRGC9Xs",
        "n": "pnXBOusEANuug6ewezb9J_...",
        "e": "AQAB"
      }
    ]
  },
  "metadata": {
    "acme_issuer": {
      "directory_url": "https://issuer.example.com/acme/directory"
    }
  }
}
```

8. Publication of the Certificates within the Federation

The X.509 Certificates issued by federation Immediate Superior Entities pertaining to one or more Federation Entity Keys in control of their Subordinates MAY publish this information by including the x5c member in each JWK contained within the matching Subordinate Statement. The contents of the published x5c member, including whether it contains a full or partial Trust Chain, and if so, to what Trust Anchor, are policy decisions out of scope for this document.

9. CSR and Certificate Fields

Depending on the Certificate Issuer’s X.509 Certificate profile, the CSR and X.509 Certificate MAY associate the X.509 Certificate to the Federation Entity by including the Entity Identifier in the X.509 Certificate.

To do so, the Issuer includes a Subject Alternative Name extension containing an otherName with a type-id of id-on-OpenIdFederationEntityId. The value of the name is an Octet String containing the UTF-8 encoding of the Entity Identifier (i.e., the URI in the corresponding openid-federation ACME Identifier from the newOrder request).

```
id-on-OpenIdFederationEntityId OBJECT IDENTIFIER ::= { id-on XXX }

OpenIdFederationEntityId ::= UTF8String
```

10. Certificate Lifecycle

The identity of the Requestor is verified through proof of possession of a private key corresponding to a public key attested within a Trust Chain. The Trust Chain has an expiration time, and its content MUST NOT be trusted past the expiration time (Section 10.2 of [OPENID-FED]).

The notBefore and notAfter fields of issued certificates MUST represent dates before the Trust Chain's expiration time. If the Requestor's newOrder request (Section 7.4 of [RFC8555]) contains notAfter or notBefore fields that make this impossible, the Certificate Issuer MUST reply with an error of type urn:ietf:params:acme:error:openIDFederationCertificateValidity.

11. Errors

This document defines two new error type URIs to be used in problem documents [RFC9457], as described in Section 6.7 of [RFC8555].

The error type urn:ietf:params:acme:error:openIDFederationEntity is used to encapsulate any OAuth error code returned while resolving OpenID Federation Entities. The title of this error type is "OpenID Federation Error". The detail member of the problem document MAY include the description of the particular OAuth error code that caused the error. The problem document for this error type SHOULD include an extension member named error_code, which MUST be set to the OAuth error code, taken from the error codes defined in Section 8.9 of [OPENID-FED].

The error type urn:ietf:params:acme:error:openIDFederationCertificateValidity is used to indicate that a Certificate could not be issued because of a mismatch between the requested period of validity and the expiration of the OpenID Federation Trust Chain.

12. Security Considerations

The openid-federation-01 challenge defined in Section 5 defends against replay attacks by malicious ACME servers because the signature in challenge responses is over an ACME key authorization, which binds the ACME account key.

The cryptographic keys in the acme_requestor metadata SHOULD NOT be reused for other purposes than signing responses to acme-federation-01 challenges. For example, the same keys SHOULD NOT be reused in the issued X.509 Certificate. If the keys are reused for other purposes, cross-protocol attacks MUST be considered.

The cryptographic keys in the acme_requestor metadata SHOULD be rotated periodically.

13. IANA Considerations

IANA is kindly asked to make the following updates to registries:

13.1. ACME Registry Group

The following updates are all assignments in the "Automated Certificate Management Environment (ACME) Protocol" registry group [IANA-ACME].

13.1.1. ACME Identifier Types

IANA is asked to add to the "ACME Identifier Types" registry, defined in Section 9.7.7 of [RFC8555], the entry below, as specified here in Section 4:

Label	Reference
openid-federation	this document

Table 1

13.1.2. ACME Validation Methods

IANA is also asked to add to the "ACME Validation Methods" registry, defined in Section 9.7.8 of [RFC8555], the entry below, as specified here in Section 5:

Label	Identifier Type	Reference
openid-federation-01	openid-federation	this document

Table 2

13.1.3. ACME Error Types

IANA is also asked to add to the "ACME Error Types" registry, defined in Section 9.7.4 of [RFC8555], the entry below, as specified here in Section 11:

Type	Description	Reference
openIDFederationEntity	An error occurred while resolving an OpenID Federation entity	this document
openIDFederationCertificateValidity	A certificate was requested whose validity period is not before the OpenID Federation Trust Chain's expiry	this document

Table 3

13.2. Assign X.509 PKIX Other Name

IANA is asked to add to the "PKIX Other Name Forms" registry (1.3.6.1.5.5.7.8 (https://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.8)) the entry below, as specified here in Section 9

Decimal	Description	Reference
TBA	id-on-OpenIdFederationEntityId	this document

Table 4

14. References

14.1. Normative References

[IANA-ACME]
IANA, "Automated Certificate Management Environment (ACME) Protocol", n.d., <https://www.iana.org/assignments/acme>.

[IANA-OAUTH]
IANA, "OAuth Parameters", n.d., <https://www.iana.org/assignments/oauth-parameters>.

[OPENID-FED]
Hedberg, R., Jones, M. B., Solberg, A. ., Bradley, J., De Marco, G., and V. Dzhuvinov, "OpenID Federation 1.0 - draft 43", 2 June 2025, <https://openid.net/specs/openid-federation-1_0-43.html>.

[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/rfc/rfc2119>.

[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/rfc/rfc2986>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/rfc/rfc4086>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[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/rfc/rfc5280>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[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/rfc/rfc8174>.

[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/rfc/rfc8555>.

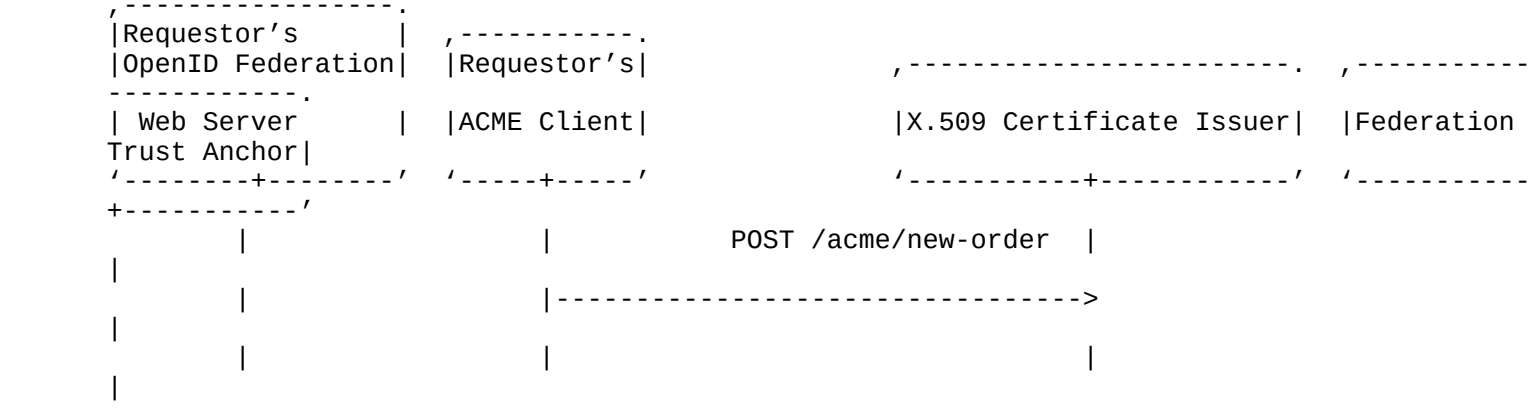
[RFC9457] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457, DOI 10.17487/RFC9457, July 2023, <https://www.rfc-editor.org/rfc/rfc9457>.

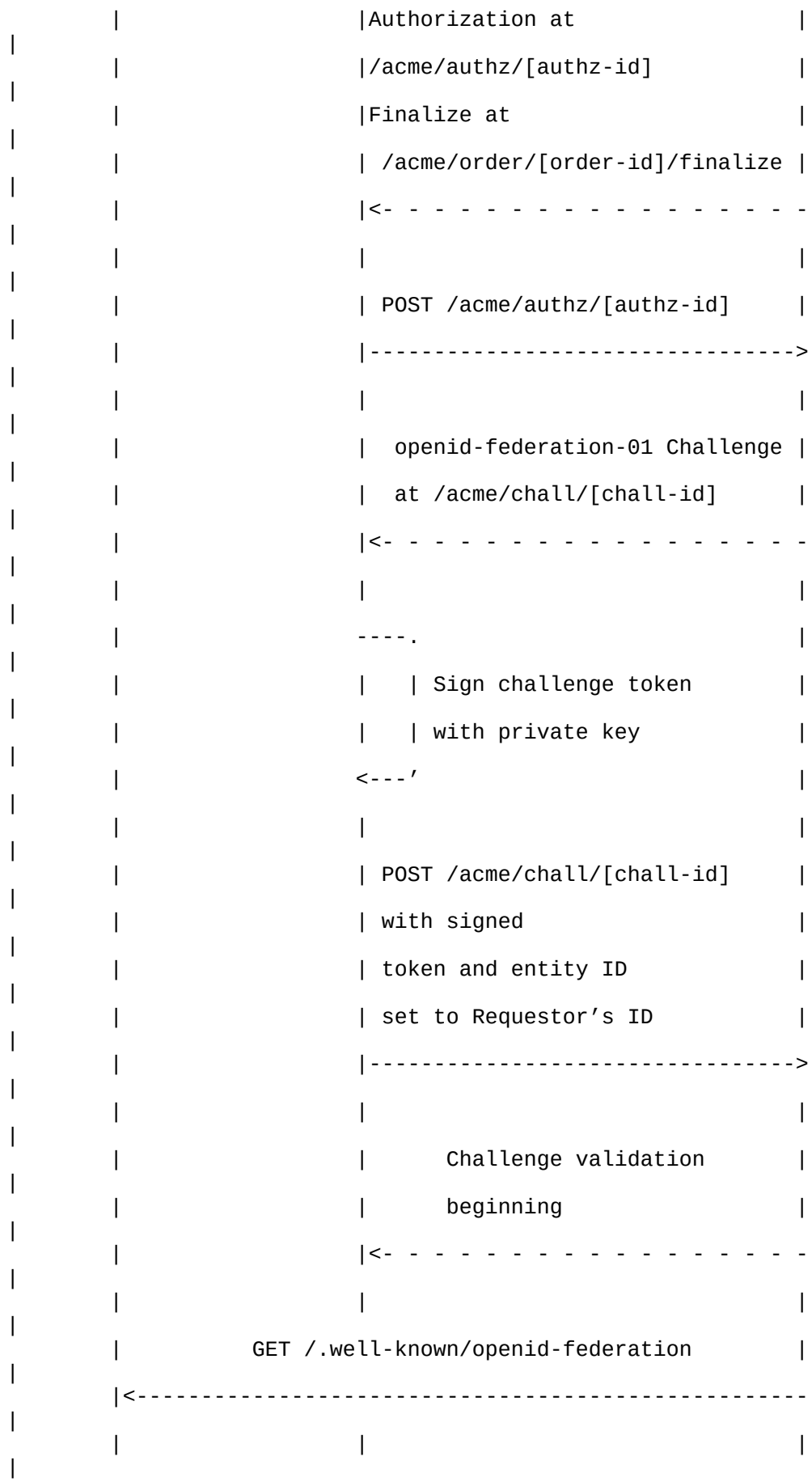
14.2. Informative References

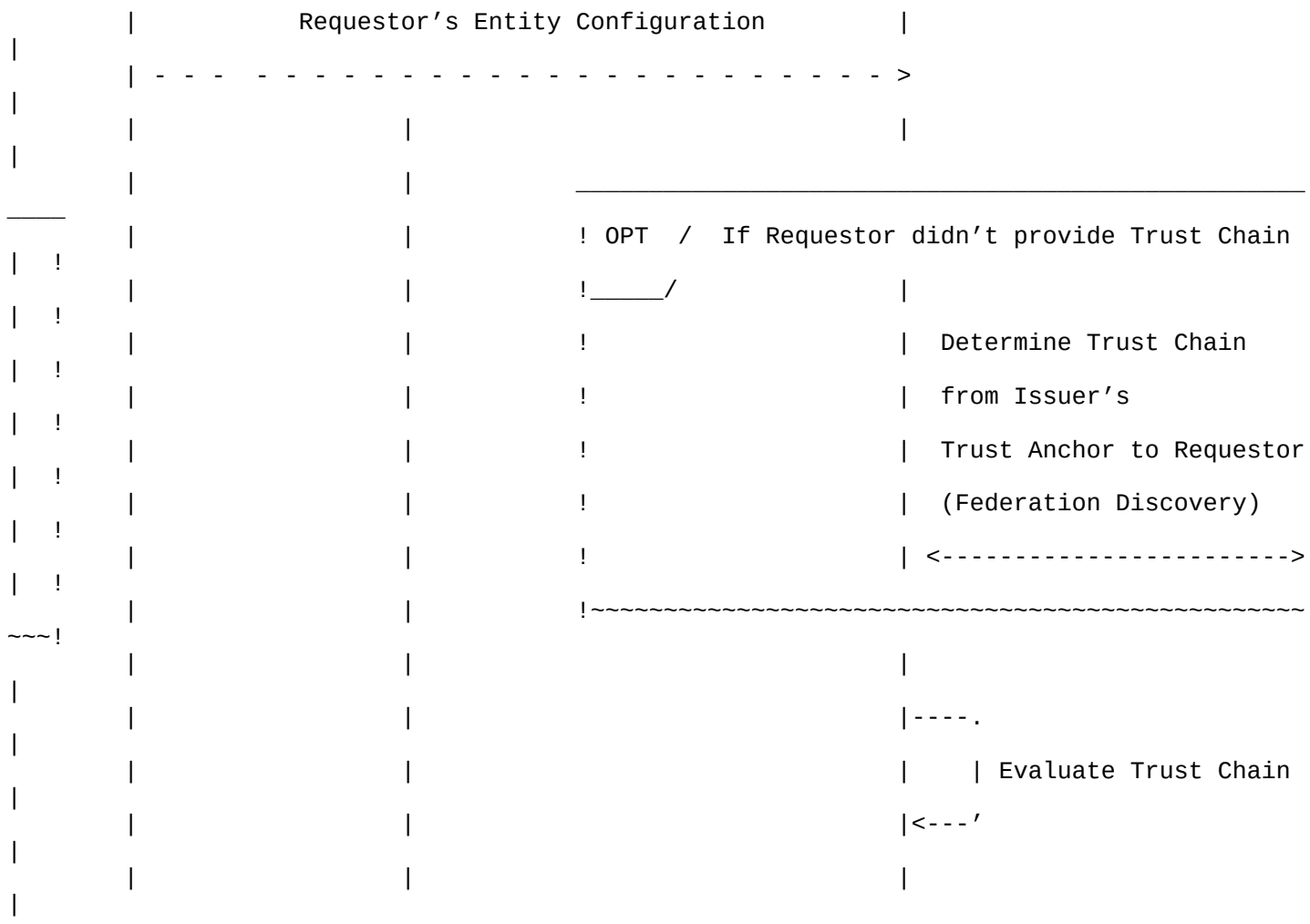
[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <https://www.rfc-editor.org/rfc/rfc2865>.

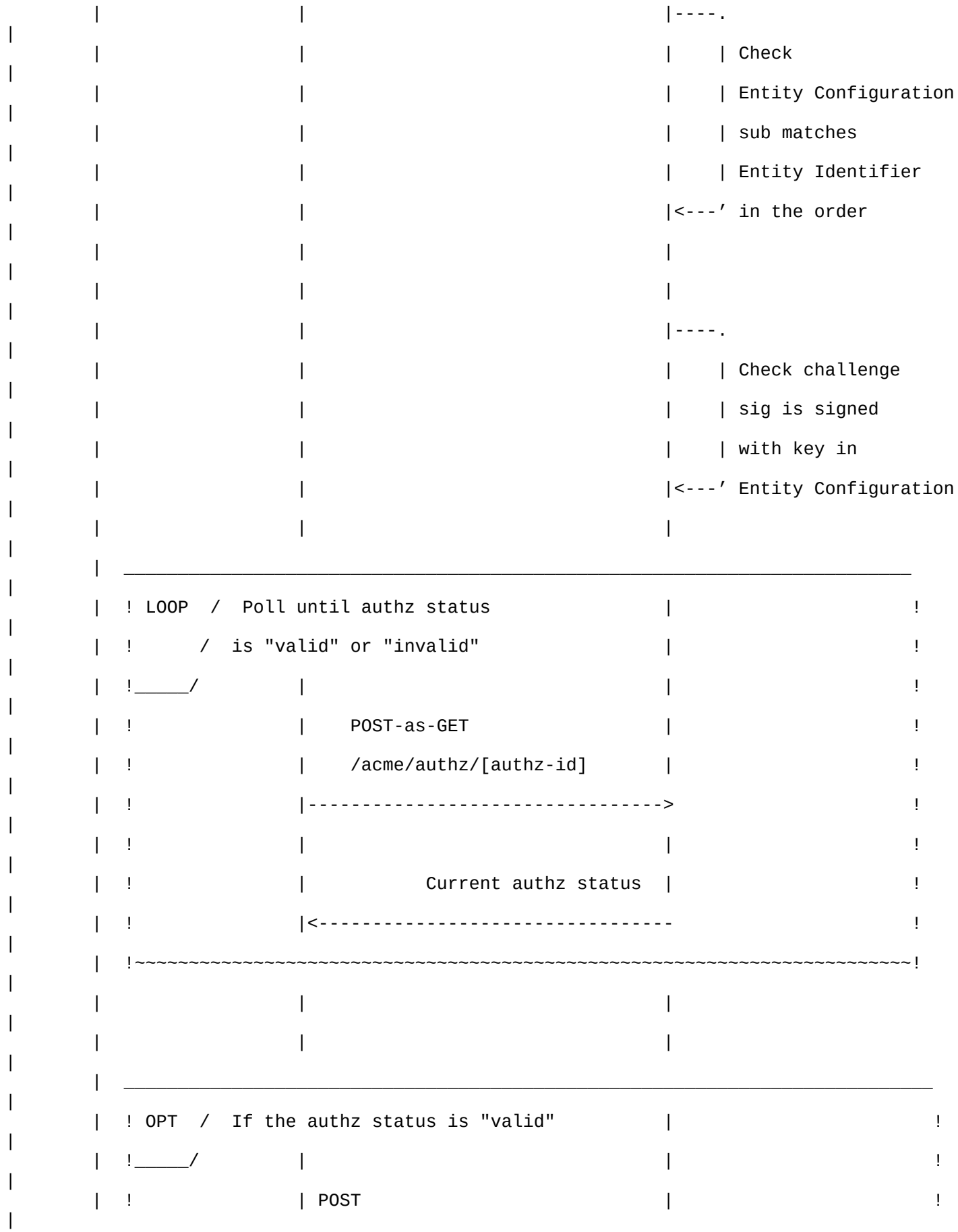
[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

Appendix A. End-to-end Issuance Flow

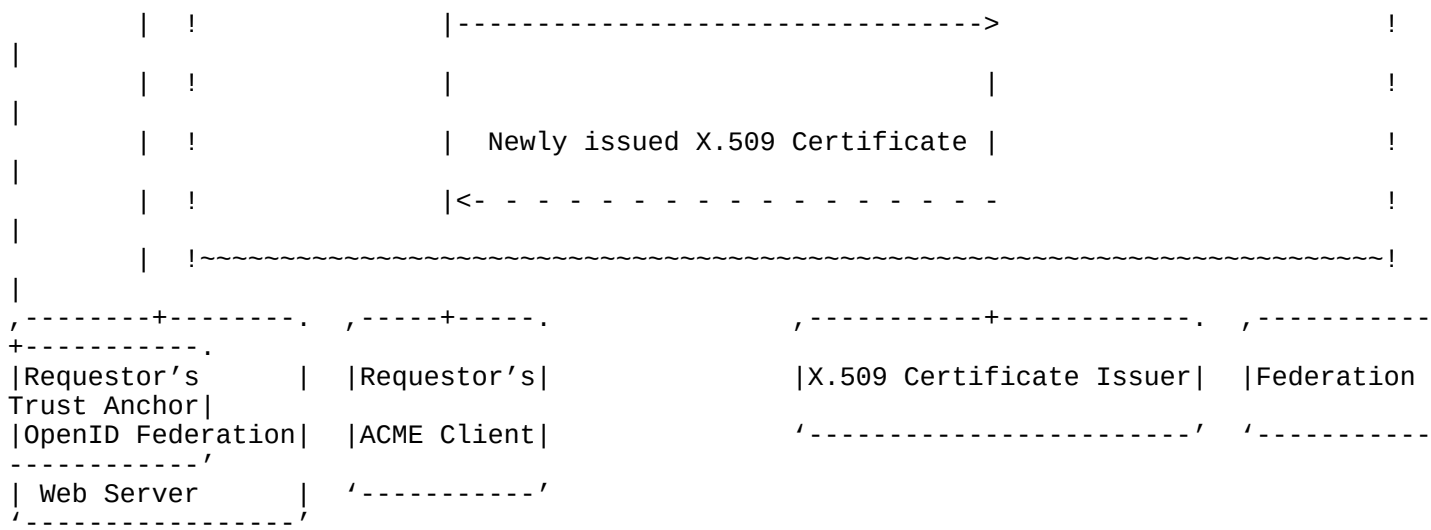
~~~ BEGIN EDNOTE ~~~

This appendix is to be removed before publication.

This section contains explanatory material that recaps a lot of RFC 8555. It is included here for the benefit of readers who are familiar with OpenID Federation but not with ACME, and want to see at a glance how the whole thing fits together.

The following diagram illustrates a successful interaction between Issuer and Requestor to retrieve an X.509 Certificate. The diagram assumes the Requestor has already discovered the Issuer, and the Requestor has already created an ACME account with the Issuer.

This section is informational and non-normative. The remainder of this document, [RFC8555] and [OPENID-FED] should be considered correct should this appendix disagree with any of them.


~~~


Authorization at
/acme/authz/[authz-id]
Finalize at
/acme/order/[order-id]/finalize
POST /acme/authz/[authz-id]
openid-federation-01 Challenge
at /acme/chall/[chall-id]
Sign challenge token
with private key
POST /acme/chall/[chall-id]
with signed
token and entity ID
set to Requestor’s ID
Challenge validation
beginning
GET /.well-known/openid-federation


Requestor's Entity Configuration
OPT / If Requestor didn't provide Trust Chain
Determine Trust Chain
from Issuer's
Trust Anchor to Requestor
(Federation Discovery)
Evaluate Trust Chain


Check
Entity Configuration
sub matches
Entity Identifier
in the order
Check challenge
sig is signed
with key in
Entity Configuration
LOOP / Poll until authz status
/ is "valid" or "invalid"
POST-as-GET
/acme/authz/[authz-id]
Current authz status
OPT / If the authz status is "valid"
POST


! | /acme/orders/[order-id]/finalize| !
! | with CSR | !
! |----------------------------------> !
! | | !
! | |----. !
! | | | Check CSR !
! | | | validity !
! | | | according to !
! | | | protocol !
! | | |<---’ and CA policy !
! | | | !
! | | | !
! | Order object with certificate | !
! | at /acme/cert/[cert-id] | !
! |<- - - - - - - - - - - - - - - - - - !
! | | !
! | POST /acme/cert/[cert-id] | !



~~~ END EDNOTE ~~~

Acknowledgments

Aaron Gable and Mike Ounsworth, for early and thorough reviews of this draft.

Contributors

David Cook
ISRG
Email: divergentdave@gmail.com

Ameer Ghani
ISRG
Email: inahga@letsencrypt.org

J.C. Jones
ISRG
Email: ietf@insufficient.coffee

Tim Geoghegan
ISRG
Email: timgeog+ietf@gmail.com

Authors' Addresses

Giuseppe De Marco
independent
Email: demarcog83@gmail.com

Brandon Pitman
~~~

Email: bran@bran.land