Automated Certificate Management Environment
Internet-Draft
Intended status: Standards Track
Expires: 14 December 2025

G. De Marco
independent
B. Pitman

12 June 2025

Automatic Certificate Management Environment (ACME) with OpenID Federation 1.0
draft-demarco-acme-openid-federation-00

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 the possession of public keys, protocol specific metadata and various 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

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 14 December 2025.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction 3
2. Target Audience and Use Cases 4
3. Terminology 4
4. Conventions and Definitions 5
5. Certificates issued using OpenID Federation 5
6. Protocol Flow 5
  6.1. Preconditions 5
  6.2. Discovery 6
  6.3. Overview 6
  6.4. Entity Configuration Metadata 9
    6.4.1. Issuer Metadata 9
    6.4.2. Requestor Metadata 10
  6.5. OpenID Federation Identifier 12
  6.6. newOrder Request 12
  6.7. OpenID Federation Challenge Type 12

6.7.1. CSR and Certificate Fields 15
7. Publication of the Certificates within the Federation 16
8. Certificate Lifecycle and Revocation 16
9. Errors 16
10. Security Considerations 16
11. IANA Considerations 17
11.1. ACME Registry Group 17
11.1.1. ACME Identifier Types 17
11.1.2. ACME Validation Methods 17
11.1.3. ACME Error Types 18
11.2. Assign X.509 PKIX Other Name 18
12. Normative References 18
Contributors 19
Authors' Addresses 20

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. X.509 Certificates can be provided to one or more organizations, without having pre-established any direct relationship or any stipulation of a contract.

In a multilateral federation, composed by 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.

Considering that a requestor is an entity requesting the issuance of an X.509 Certificate to a server and the issuer is the ACME server that validates the entitlements of the requestor before issuing the X.509 Certificate, this specification defines how ACME and OpenID Federation 1.0 can be integrated to allow efficient issuance of X.509 Certificates to a requestor via the introduction of a new ACME challenge type. The new challenge type extends the ACME protocol in the following ways:

* It associates a cryptographic key with an OpenID Entity, rather than a domain, since the authentication and authorization of the requestor is asserted with OpenID Federation 1.0.

* It defines how to use and validate a basic OpenID Federation component, called Entity Configuration, that is a signed JWT published in a well-known resource (/.well-known/openid-federation).

* It defines how the OpenID Federation Subordinate Statements can be used for the publication of the X.509 Certificates, by a Superior Entity, that were previously issued with ACME.

* It extends the ACME newOrder resource, as defined in Section 7.4 of [RFC8555], defining a new payload identifier type called openid-federation.

2. Target Audience and Use Cases

The audience of the document are the multilateral federations that require automatic issuance of X.509 Certificates using an infrastructure of trust based on OpenID Federation 1.0.

This specification can be implemented by:

* Federation Entities that join to a federation staging area using HTTP only transport to attest themselves as trustworthy, and then retrieve X.509 Certificates for their official HTTPS Federation Entity ID.

* Federation Entities that want to ask for and obtain X.509 Certificate for use in other protocols.

3. Terminology

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

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: 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.

4. 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.

5. Certificates issued using OpenID Federation

The Certificate Issuer establishes the authorization of a Federation Entity to obtain X.509 Certificates for the identifier configured in the Requestor’s Entity Configuration.

The Certificate Issuer establishes if a Federation Entity is eligible to obtain X.509 Certificates for the identifier configured in the Requestor’s Entity Configuration.

The cryptographic keys published within the Requestor’s Entity Configuration are used to satisfy the Certificate Issuer’s challenge.

The protocol assumes the following discovery preconditions are met. The Issuer has the guarantee that:

1. The Requestor controls the private key related to the public part published in its Entity Configuration.

2. The Requestor controls its identifier, having published the Entity Configuration.

6. Protocol Flow

This section presents the protocol flow. The protocol flow is subdivided in the following phases:

* *Discovery*, the Requestor obtains the available CAs within a federation, inspecting the ACME issuer entity types.

* *Order request*, the Requestor requests a X.509 Certificate to a Certificate Issuer using the ACME protocol.

6.1. Preconditions

The protocol requires the following preconditions are met.

1. The Requestor and the Issuer MUST publish their Entity Configuration as defined in Section 9 of [OPENID-FED].

2. The Requestor and the Issuer MUST be able to establish a Trust Chain to each other, as defined in Section 4 of [OPENID-FED], from their respective Trust Anchors.

3. The Issuer MUST implement an ACME server, extended according to this document.

4. The Requestor MUST publish the entity type acme_requestor in its Entity Configuration, according to Section 6.4.2.

5. The Issuer MUST publish the entity type acme_issuer in its Entity Configuration, according to Section 6.4.1.

6.2. 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 federation. Requestors that use discovery MAY select any entity with an entity type of acme_issuer, or they may additionally require that such entities have a valid Trust Mark with a particular Trust Mark Identifier.

6.3. Overview

1. The Requestor checks if its superior Federation Entity supports the ACME protocol for OpenID Federation 1.0. If not, the Requestor starts the discovery process to find Issuers within the federation.

2. The Requestor requests and obtains a new nonce from the Certificate Issuer, by sending a HTTP HEAD request to the Issuer's newNonce resource, as described in Section 7.2 of [RFC8555].

3. The Requestor creates an ACME Account with the Issuer, as described in Section 7.3 of [RFC8555].

4. The Requestor begins the X.509 Certificate issuance process by sending a HTTP POST request to the Certificate Issuer's newOrder resource, as described in Section 6.6, and follows the remainder of the ACME protocol as specified in [RFC8555], using the new challenge defined in Section 6.7.

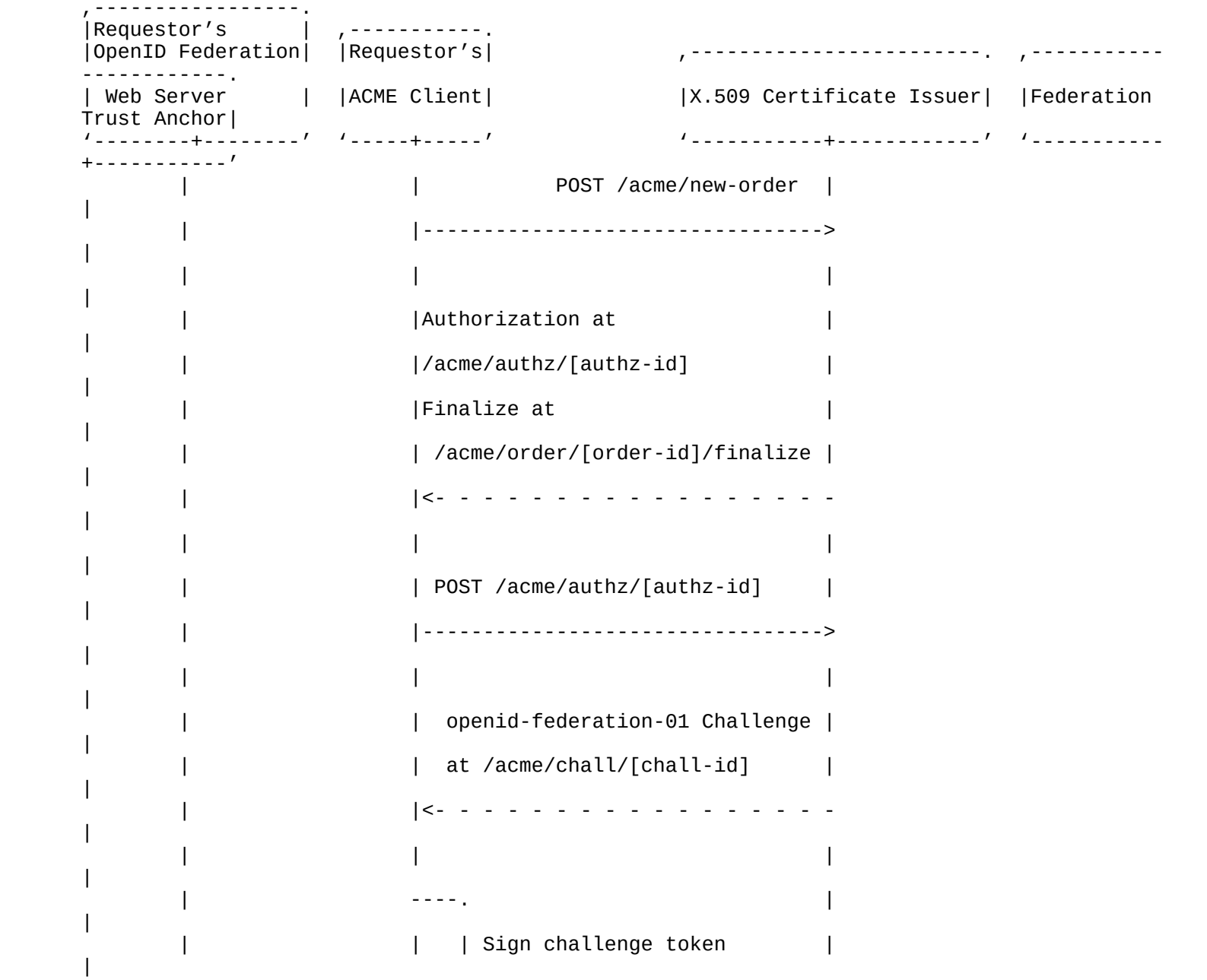
5. The Certificate Issuer evaluates the trust to the Requestor by checking if it is part of its federation. If not the CSR request MUST be rejected, with error type urn:ietf:params:acme:error:openIDFederationEntity, and an error code of invalid_trust_chain (Section 8.9 of [OPENID-FED]).

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

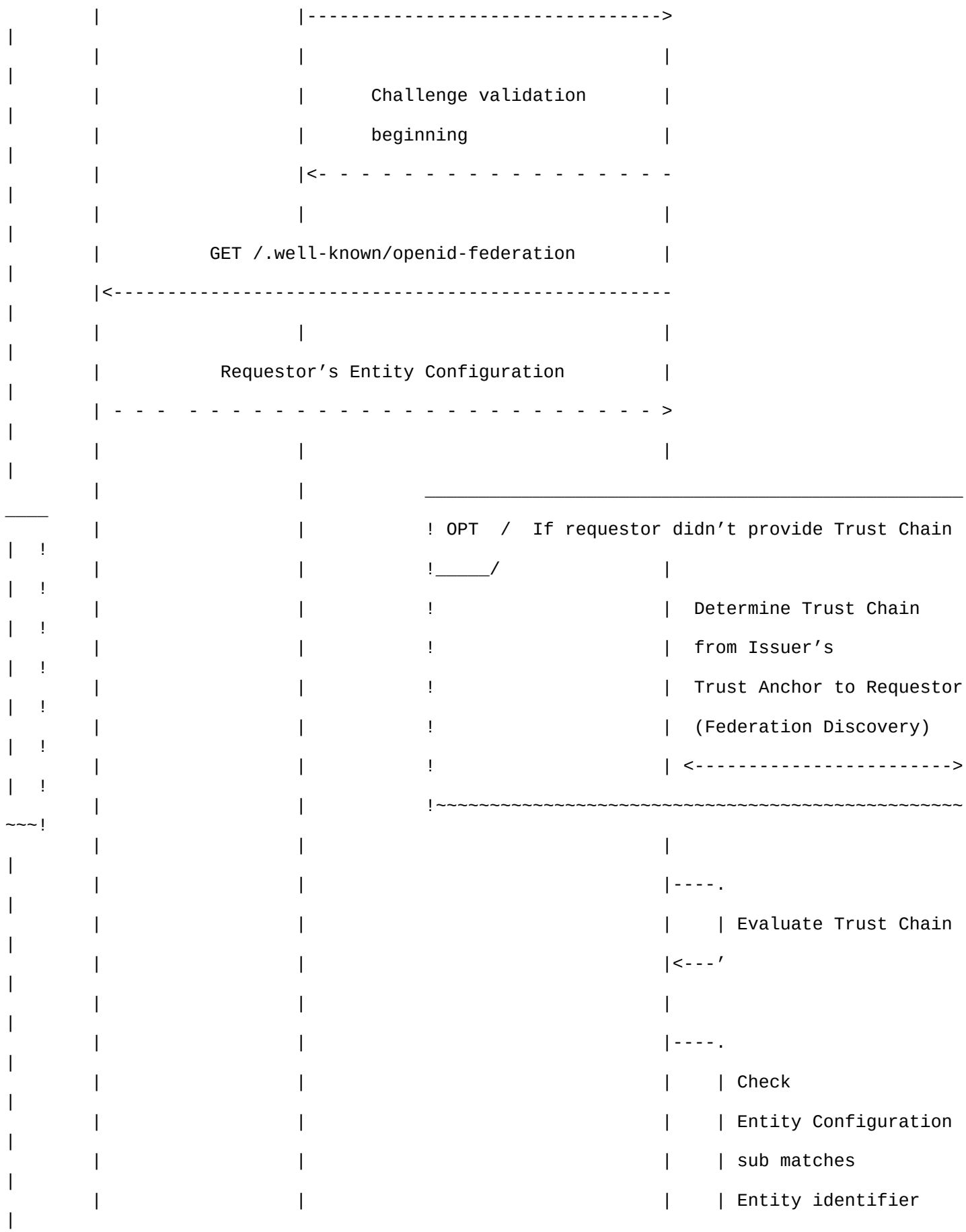
* The Requestor adds the Trust Chain JWT header parameter related to itself, as described in Section 4.3 of [OPENID-FED]. This option is RECOMMENDED since it reduces the effort of the Certificate Issuer in evaluating the trust to the Requestor.

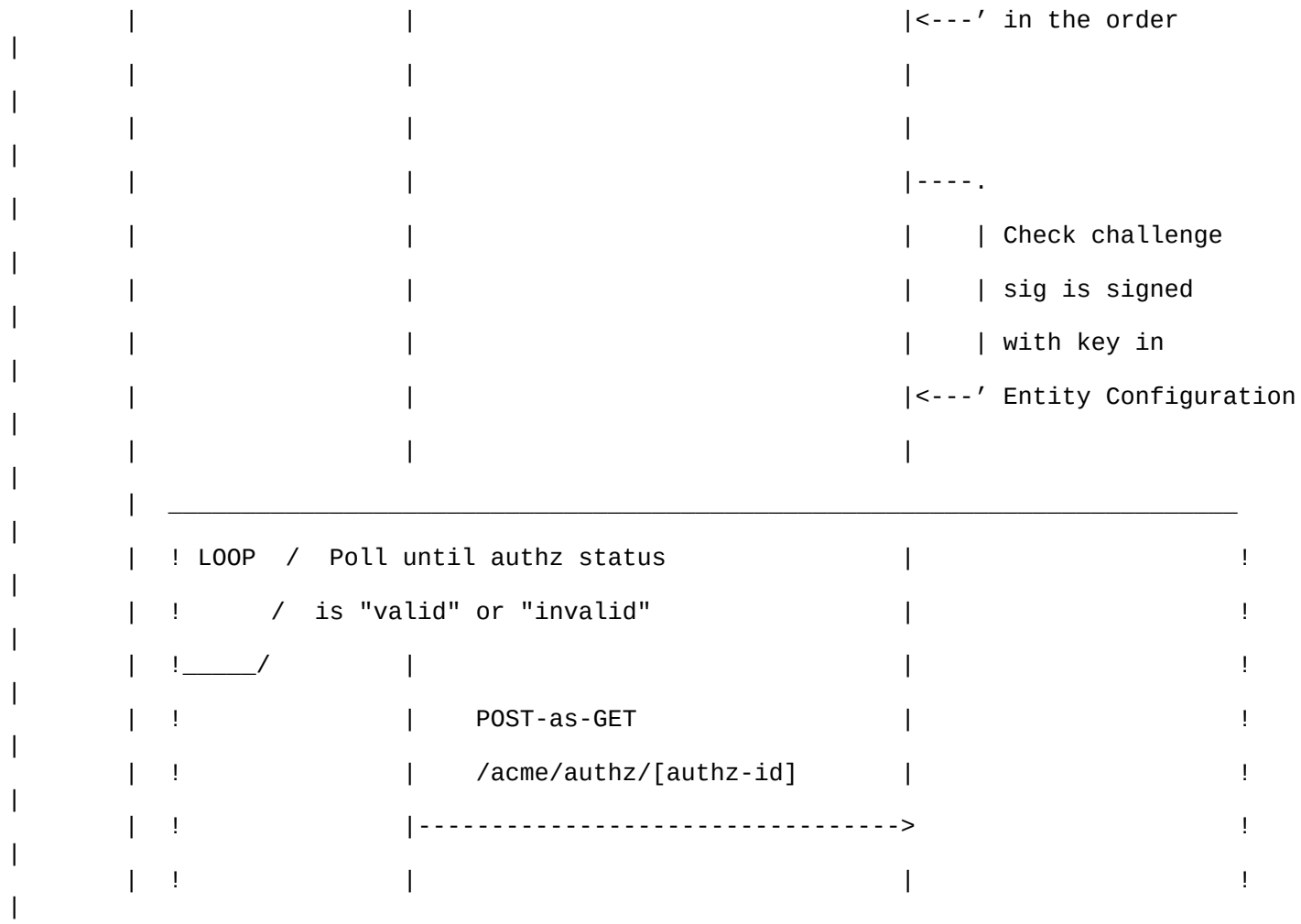
* The Requestor doesn't add the Trust Chain in the request. The Certificate Issuer MUST start Federation Entity Discovery as described in Section 9 of [OPENID-FED] to obtain the Trust Chain related to the Requestor.

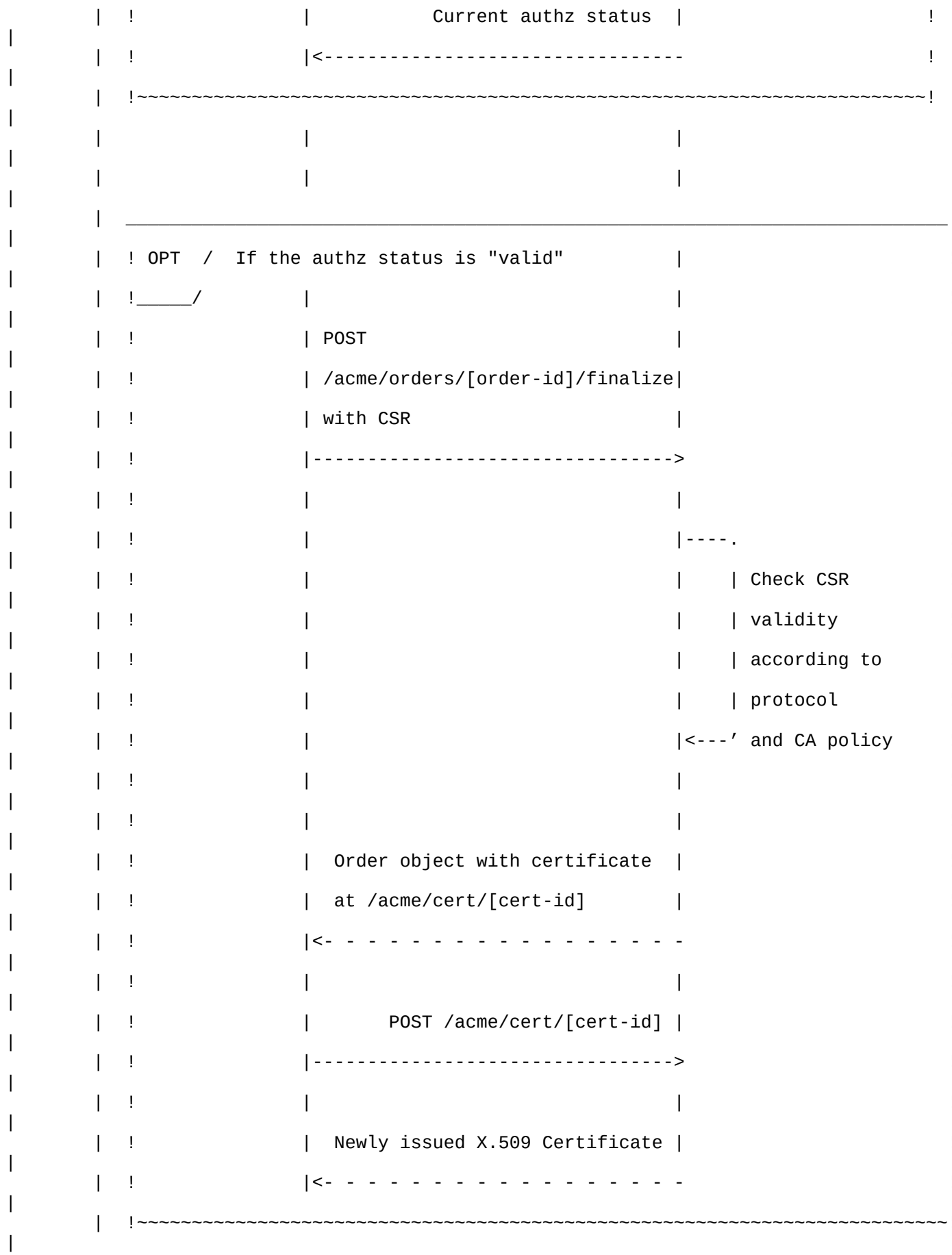
The following diagram illustrates a successful interaction between Issuer and Requestor to retrieve an X.509 Certificate. The diagram assumes the Requestor has already discovered the Issuer, and the Requestor has already created an ACME account with the Issuer.

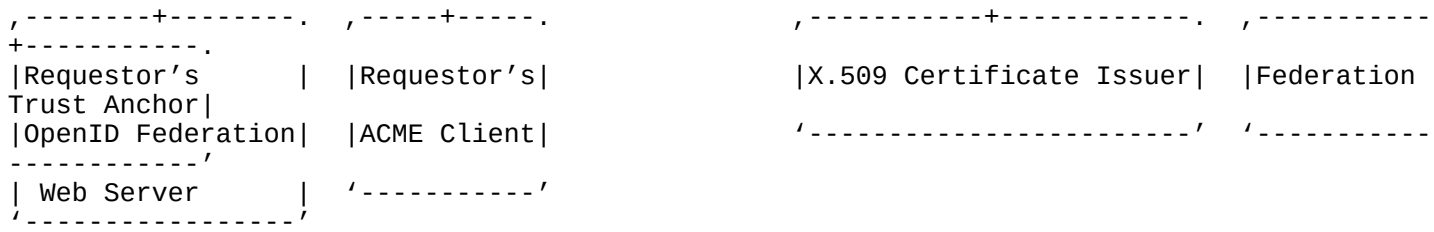



			with private key
	<---’
		POST /acme/chall/[chall-id]
		with signed
		token and entity ID
		set to Requestor’s ID


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]
Newly issued X.509 Certificate



6.4. Entity Configuration Metadata

This section describes the metadata a Requestor and Issuer MUST publish in their respective Entity Configurations.

6.4.1. 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"
    }
  }
}
```

6.4.2. Requestor 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.

6.5. OpenID Federation 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].

6.6. newOrder Request

The Requestor begins certificate issuance by sending a HTTP POST request to the Issuer's newOrder resource, as specified in Section 7.4 of [RFC8555].

A non-normative example of an ACME newOrder request:

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

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

6.7. 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.

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 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 issue X.509 Certificates for any identifiers except openid-federation identifiers.

The openid-federation identifier MUST NOT be validated except by the openid-federation-01 challenge.

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 6.4.2. 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 Entity Configuration of the Trust Chain subject MUST contain acme_requestor metadata that is valid under the Trust Chain's resolved metadata policy (Section 6.1 of [OPENID-FED]) and 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 verifies:

* That the Requestor's acme_requestor metadata is valid under the Trust Chain's resolved metadata policy (Section 6.1 of [OPENID-FED]).

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

* That the sig field of the payload is the compact JSON serialization of a JWS signing the key authorization, signed with a key from the Requestor's acme_requestor metadata (Section 6.4.2) whose kid matches the kid claim in the challenge response.

If all of the above verifications 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].

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"
}
```

6.7.1. 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 ID 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 ID (i.e., the URI in the corresponding openid-federation identifier from the newOrder request).

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

OpenIdFederationEntityId ::= UTF8String
```

7. 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.

8. Certificate Lifecycle and Revocation

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. It is up to the Certificate Issuer to decide the expiration time of the X.509 Certificate. In some cases, and when required, it MAY be set to match the expiration of the Trust Chain.

A Requestor SHOULD request the revocation of its X.509 Certificate when the related cryptographic material is revoked. The Requestor SHOULD publish the revoked or expired cryptographic keys in the Federation Historical Key Registry.

The X.509 Certificate revocation request is defined in Section 7.6 of [RFC8555].

9. Errors

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

The error type urn:ietf:params:acme:error:openIDFederationEntity can be 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. The error_code member SHOULD be set to the OAuth error code, taken from the IANA "OAuth Extensions Error Registry" [IANA-OAUTH].

10. Security Considerations

The openid-federation-01 challenge defined in Section 6.7 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.

11. IANA Considerations

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

11.1. ACME Registry Group

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

11.1.1. 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 6.5:

Label	Reference
openid-federation	this document

Table 1

11.1.2. 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 6.7:

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

Table 2

11.1.3. 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 9:

Type	Description	Reference
openIDFederationEntity	An error occurred while resolving an OpenID Federation entity	this document

Table 3

11.2. 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 6.7.1

Decimal	Description	Reference
TBA	id-on-OpenIdFederationEntityId	this document

Table 4

12. 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 41", 4 December 2024, <https://openid.net/specs/openid-federation-1_0-41.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>.

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