Stream: Internet Engineering Task Force (IETF)
RFC: 9115
Category: Standards Track
Published: August 2021
ISSN: 2070-1721
Authors: Y. Sheffer, *Intuit*; D. López, *Telefonica I+D*; A. Pastor Perales, *Telefonica I+D*; T. Fossati, *ARM*

# RFC 9115
# An Automatic Certificate Management Environment (ACME) Profile for Generating Delegated Certificates

## Abstract

This document defines a profile of the Automatic Certificate Management Environment (ACME) protocol by which the holder of an identifier (e.g., a domain name) can allow a third party to obtain an X.509 certificate such that the certificate subject is the delegated identifier while the certified public key corresponds to a private key controlled by the third party. A primary use case is that of a Content Delivery Network (CDN), the third party, terminating TLS sessions on behalf of a content provider (the holder of a domain name). The presented mechanism allows the holder of the identifier to retain control over the delegation and revoke it at any time. Importantly, this mechanism does not require any modification to the deployed TLS clients and servers.

## Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc9115.

## Copyright Notice

## Table of Contents

# 1. Introduction

This document is related to [RFC8739], in that some important use cases require both documents to be implemented. To avoid duplication, we give here a bare-bones description of the motivation for this solution. For more details, please refer to the introductory sections of [RFC8739].

An Identifier Owner (IdO) has agreements in place with one or more Name Delegation Consumer (NDC) to use and attest its identity.

In the primary use case, the IdO is a content provider, and we consider a Content Delivery Network (CDN) provider contracted to serve the content over HTTPS. The CDN terminates the HTTPS connection at one of its edge cache servers and needs to present its clients (browsers, mobile apps, set-top boxes) a certificate whose name matches the domain name of the URL that is

requested, i.e., that of the IdO. Understandably, some IdOs may balk at sharing their long-term private keys with another organization; equally, delegates would rather not have to handle other parties' long-term secrets. Other relevant use cases are discussed in Section 5.

This document describes a profile of the ACME protocol [RFC8555] that allows the NDC to request from the IdO, acting as a profiled ACME server, a certificate for a delegated identity -- i.e., one belonging to the IdO. The IdO then uses the ACME protocol (with the extensions described in [RFC8739]) to request issuance of a Short-Term, Automatically Renewed (STAR) certificate for the same delegated identity. The generated short-term certificate is automatically renewed by the ACME Certification Authority (CA), is periodically fetched by the NDC, and is used to terminate HTTPS connections in lieu of the IdO. The IdO can end the delegation at any time by simply instructing the CA to stop the automatic renewal and letting the certificate expire shortly thereafter.

While the primary use case we address is a delegation of STAR certificates, the mechanism proposed here also accommodates long-lived certificates managed with the ACME protocol. The most noticeable difference between long-lived and STAR certificates is the way the termination of the delegation is managed. In the case of long-lived certificates, the IdO uses the `revokeCert` URL exposed by the CA and waits for the explicit revocation based on the Certificate Revocation List (CRL) and Online Certificate Status Protocol (OCSP) to propagate to the relying parties.

In case the delegated identity is a domain name, this document also provides a way for the NDC to inform the IdO about the CNAME mappings that need to be installed in the IdO's DNS zone to enable the aliasing of the delegated name, thus allowing the complete name delegation workflow to be handled using a single interface.

We note that other standardization efforts address the problem of certificate delegation for TLS connections, specifically [TLS-SUBCERTS] and [MGLT-LURK-TLS13]. The former extends the TLS certificate chain with a customer-owned signing certificate; the latter separates the server's private key into a dedicated, more-secure component. Compared to these other approaches, the current document does not require changes to the TLS network stack of the client or the server, nor does it introduce additional latency to the TLS connection.

## 1.1. Terminology

IdO
: Identifier Owner, the holder (current owner) of an identifier (e.g., a domain name) that needs to be delegated. Depending on the context, the term IdO may also be used to designate the (profiled) ACME server deployed by the Identifier Owner or the ACME client used by the Identifier Owner to interact with the CA.

NDC
: Name Delegation Consumer, the entity to which the domain name is delegated for a limited time. This is a CDN in the primary use case (in fact, readers may note the similarity of the two abbreviations). Depending on the context, the term NDC may also be used to designate the (profiled) ACME client used by the Name Delegation Consumer.

CDN
: Content Delivery Network, a widely distributed network that serves the domain's web content to a wide audience at high performance.

STAR Short-Term, Automatically Renewed, as applied to X.509 certificates.

ACME Automated Certificate Management Environment, a certificate management protocol [RFC8555].

CA Certification Authority, specifically one that implements the ACME protocol. In this document, the term is synonymous with "ACME server deployed by the Certification Authority".

CSR Certificate Signing Request, specifically a PKCS#10 [RFC2986] Certificate Signing Request, as supported by ACME.

FQDN Fully Qualified Domain Name.

### 1.2. Conventions Used in This Document

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.

## 2. Protocol Flow

This section presents the protocol flow. For completeness, we include the ACME profile proposed in this document as well as the ACME STAR protocol described in [RFC8739].

### 2.1. Preconditions

The protocol assumes the following preconditions are met:

- The IdO exposes an ACME server interface to the NDC(s) comprising the account management interface.
- The NDC has registered an ACME account with the IdO.
- The NDC and IdO have agreed on a "CSR template" to use, including at a minimum: subject name (e.g., `abc.ido.example`), requested algorithms and key length, key usage, and extensions. The NDC will use this template for every CSR created under the same delegation.
- The IdO has registered an ACME account with the Certification Authority (CA).

Note that even if the IdO implements the ACME server role, it is not acting as a CA; in fact, from the point of view of the certificate issuance process, the IdO only works as a "policing" forwarder of the NDC's key pair and is responsible for completing the identity verification process towards the CA.

## 2.2. Overview

For clarity, the protocol overview presented here covers the main use case of this protocol, namely delegation of STAR certificates. Protocol behavior for non-STAR certificates is similar, and the detailed differences are listed in the following sections.

The interaction between the NDC and the IdO is governed by the profiled ACME workflow detailed in Section 2.3. The interaction between the IdO and the CA is ruled by ACME [RFC8555], ACME STAR [RFC8739], and any other ACME extension that applies (e.g., [TOKEN-TNAUTHLIST] for Secure Telephone Identity Revisited (STIR)).

The outline of the combined protocol for STAR certificates is as follows (Figure 1):

- NDC sends an Order1 for the delegated identifier to IdO.
- IdO creates an Order1 resource in state `ready` with a `finalize` URL.
- NDC immediately sends a `finalize` request (which includes the CSR) to the IdO.
- IdO verifies the CSR according to the agreed upon CSR template.
- If the CSR verification fails, Order1 is moved to an `invalid` state and everything stops.
- If the CSR verification is successful, IdO moves Order1 to state `processing` and sends a new Order2 (using its own account) for the delegated identifier to the CA.
- If the ACME STAR protocol fails, Order2 moves to `invalid`, and the same state is reflected in Order1 (i.e., the NDC Order).
- If the ACME STAR run is successful (i.e., Order2 is `valid`), IdO copies the `star-certificate` URL from Order2 to Order1 and updates the Order1 state to `valid`.

The NDC can now download, install, and use the short-term certificate bearing the name delegated by the IdO. The STAR certificate can be used until it expires, at which time the NDC is guaranteed to find a new certificate it can download, install, and use. This continues with subsequent certificates until either Order1 expires or the IdO decides to cancel the automatic renewal process with the CA.

Note that the interactive identifier authorization phase described in Section 7.5 of [RFC8555] is suppressed on the NDC-IdO side because the delegated identity contained in the CSR presented to the IdO is validated against the configured CSR template (Section 4.1). Therefore, the NDC sends the `finalize` request, including the CSR, to the IdO immediately after Order1 has been acknowledged. The IdO **SHALL** buffer a (valid) CSR until the Validation phase completes successfully.

Also note that the successful negotiation of the unauthenticated GET (Section 3.4 of [RFC8739]) is required in order to allow the NDC to access the `star-certificate` URL on the CA.

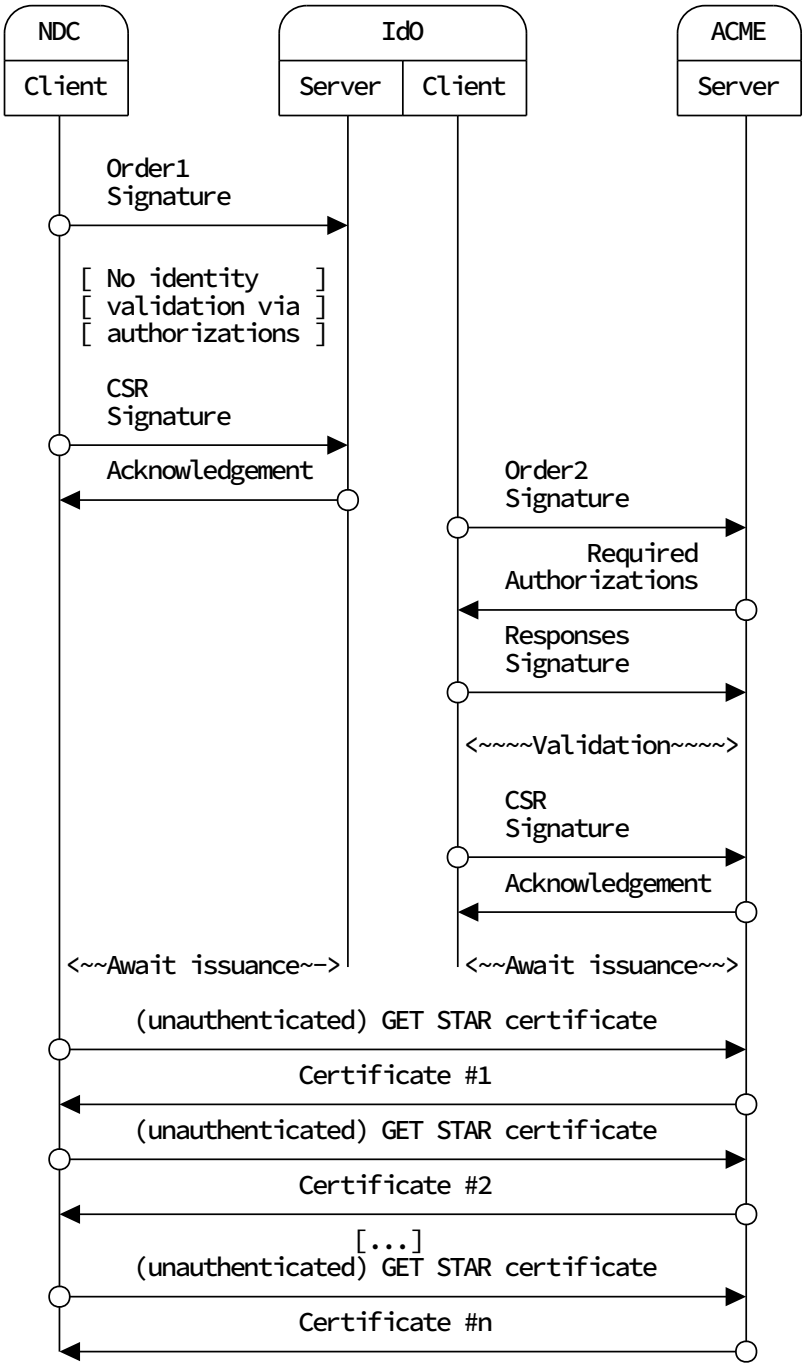


*Figure 1: End-to-End STAR Delegation Flow*

## 2.3. Delegated Identity Profile

This section defines a profile of the ACME protocol to be used between the NDC and IdO.

### 2.3.1. Delegation Configuration

The IdO must be preconfigured to recognize one or more NDCs and present them with details about certificate delegations that apply to each one.

#### 2.3.1.1. Account Object Extensions

An NDC identifies itself to the IdO as an ACME account. The IdO can delegate multiple names to an NDC, and these configurations are described through `delegation` objects associated with the NDC's account object on the IdO.

As shown in Figure 2, the ACME account resource on the IdO is extended with a new `delegations` attribute:

delegations (required, string): A URL from which a list of delegations configured for this account (Section 2.3.1.3) can be fetched via a POST-as-GET request.

```
{
  "status": "valid",
  "contact": [
    "mailto:delegation-admin@ido.example"
  ],
  "termsOfServiceAgreed": true,
  "orders": "https://example.com/acme/orders/saHpfB",
  "delegations": "https://acme.ido.example/acme/delegations/adFqoz"
}
```

*Figure 2: Example Account Object with Delegations*

#### 2.3.1.2. Delegation Lists

Each account object includes a `delegations` URL from which a list of delegation configurations created by the IdO can be fetched via a POST-as-GET request. The result of the request **MUST** be a JSON object whose `delegations` field is an array of URLs, each identifying a delegation

configuration made available to the NDC account (Section 2.3.1.3). The server **MAY** return an incomplete list, along with a `Link` header field with a `next` link relation indicating where further entries can be acquired.

```
HTTP/1.1 200 OK
Content-Type: application/json
Link: <https://acme.ido.example/acme/directory>;rel="index"
Link: <https://acme.ido.example/acme/delegations/adFqoz?/
      cursor=2>;rel="next"

{
  "delegations": [
    "https://acme.ido.example/acme/delegation/ogfr8EcolOT",
    "https://acme.ido.example/acme/delegation/wSi5Lbb61E4",
    /* more URLs not shown for example brevity */
    "https://acme.ido.example/acme/delegation/gm0wfLYHBen"
  ]
}
```

Note that in the figure above, https://acme.ido.example/acme/delegations/adFqoz?cursor=2 includes a line break for the sake of presentation.

#### 2.3.1.3. Delegation Objects

This profile extends the ACME resource model with a new read-only `delegation` object that represents a delegation configuration that applies to a given NDC.

A `delegation` object contains the CSR template (see Section 4) that applies to that delegation and, optionally, any related CNAME mapping for the delegated identifiers. Its structure is as follows:

csr-template (required, object): CSR template, as defined in Section 4.

cname-map (optional, object): A map of FQDN pairs. In each pair, the name is the delegated identifier; the value is the corresponding NDC name that is aliased in the IdO's zone file to redirect the resolvers to the delegated entity. Both names and values **MUST** be FQDNs with a terminating '.'. This field is only meaningful for identifiers of type `dns`.

An example `delegation` object in JSON format is shown in Figure 3.

```
{
  "csr-template": {
    "keyTypes": [
      {
        "PublicKeyType": "id-ecPublicKey",
        "namedCurve": "secp256r1",
        "SignatureType": "ecdsa-with-SHA256"
      }
    ],
    "subject": {
      "country": "CA",
      "stateOrProvince": "**",
      "locality": "**"
    },
    "extensions": {
      "subjectAltName": {
        "DNS": [
          "abc.ido.example"
        ]
      },
      "keyUsage": [
        "digitalSignature"
      ],
      "extendedKeyUsage": [
        "serverAuth"
      ]
    }
  },
  "cname-map": {
    "abc.ido.example.": "abc.ndc.example."
  }
}
```

*Figure 3: Example Delegation Configuration Object*

In order to indicate which specific delegation applies to the requested certificate, a new `delegation` attribute is added to the order object on the NDC-IdO side (see Figures 4 and 7). The value of this attribute is the URL pointing to the delegation configuration object that is to be used for this certificate request. If the `delegation` attribute in the order object contains a URL that does not correspond to a configuration available to the requesting ACME account, the IdO **MUST** return an error response with status code 403 (Forbidden), providing a problem document [RFC7807] with type `urn:ietf:params:acme:error:unknownDelegation`.

#### 2.3.2. Order Object Transmitted from NDC to IdO and to ACME Server (STAR)

If the delegation is for a STAR certificate, the request object created by the NDC:

- **MUST** have a `delegation` attribute indicating the preconfigured delegation that applies to this Order;
- **MUST** have entries in the `identifiers` field for each delegated name present in the configuration;
- **MUST NOT** contain the `notBefore` and `notAfter` fields; and

- **MUST** contain an `auto-renewal` object and, inside it, the fields listed in Section 3.1.1 of [RFC8739]. In particular, the `allow-certificate-get` attribute **MUST** be present and set to true.

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

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://acme.ido.example/acme/acct/evOfKhNU60wg",
    "nonce": "Alc00Ap6Rt7GMkEl3L1JX5",
    "url": "https://acme.ido.example/acme/new-order"
  }),
  "payload": base64url({
    "identifiers": [
      {
        "type": "dns",
        "value": "abc.ido.example"
      }
    ],
    "auto-renewal": {
      "end-date": "2021-04-20T00:00:00Z",
      "lifetime": 345600,          // 4 days
      "allow-certificate-get": true
    },
    "delegation":
      "https://acme.ido.example/acme/delegation/gm0wfLYHBen"
  }),
  "signature": "g454e3hdBlkT4AEw...nKePnUyZTjGtXZ6H"
}
```

*Figure 4: New STAR Order from NDC*

The order object that is created on the IdO:

- **MUST** start in the `ready` state;
- **MUST** contain an `authorizations` array with zero elements;
- **MUST** contain the indicated `delegation` configuration;
- **MUST** contain the indicated `auto-renewal` settings; and
- **MUST NOT** contain the `notBefore` and `notAfter` fields.

```
{
  "status": "ready",
  "expires": "2021-05-01T00:00:00Z",

  "identifiers": [
   {
     "type": "dns",
     "value": "abc.ido.example"
   }
  ],

  "auto-renewal": {
    "end-date": "2021-04-20T00:00:00Z",
    "lifetime": 345600,
    "allow-certificate-get": true
  },

  "delegation":
    "https://acme.ido.example/acme/delegation/gm0wfLYHBen",

  "authorizations": [],

  "finalize": "https://acme.ido.example/acme/order/TO8rfgo/finalize"
}
```

*Figure 5: STAR Order Resource Created on IdO*

The Order is then finalized by the NDC supplying the CSR containing the delegated identifiers. The IdO checks the provided CSR against the template contained in the `delegation` object that applies to this request, as described in Section 4.1. If the CSR fails validation for any of the identifiers, the IdO **MUST** return an error response with status code 403 (Forbidden) and an appropriate type, e.g., `rejectedIdentifier` or `badCSR`. The error response **SHOULD** contain subproblems (Section 6.7.1 of [RFC8555]) for each failed identifier. If the CSR is successfully validated, the order object status moves to `processing` and the twin ACME protocol instance is initiated on the IdO-CA side.

The request object created by the IdO:

- **MUST** copy the identifiers sent by the NDC;
- **MUST** strip the `delegation` attribute; and
- **MUST** carry a copy of the `auto-renewal` object sent by the NDC.

When the identifiers' authorization has been successfully completed and the certificate has been issued by the CA, the IdO:

- **MUST** move its Order resource status to `valid` and
- **MUST** copy the `star-certificate` field from the STAR Order returned by the CA into its Order resource. When dereferenced, the `star-certificate` URL includes (via the `Cert-Not-Before` and `Cert-Not-After` HTTP header fields) the renewal timers needed by the NDC to inform its certificate reload logic.

```
{
  "status": "valid",
  "expires": "2021-05-01T00:00:00Z",

  "identifiers": [
   {
     "type": "dns",
     "value": "abc.ido.example"
   }
  ],

  "auto-renewal": {
    "end-date": "2021-04-20T00:00:00Z",
    "lifetime": 345600,
    "allow-certificate-get": true
  },

  "delegation":
    "https://acme.ido.example/acme/delegation/gm0wfLYHBen",

  "authorizations": [],

  "finalize": "https://acme.ido.example/acme/order/TO8rfgo/finalize",

  "star-certificate": "https://acme.ca.example/acme/order/yTr23sSDg9"
}
```

*Figure 6: STAR Order Resource Updated on IdO*

This delegation protocol is predicated on the NDC being able to fetch certificates periodically using an unauthenticated HTTP GET, since, in general, the NDC does not possess an account on the CA; as a consequence, it cannot issue the standard POST-as-GET ACME request. Therefore, before forwarding the Order request to the CA, the IdO **SHOULD** ensure that the selected CA supports unauthenticated GET by inspecting the relevant settings in the CA's directory object, as per Section 3.4 of [RFC8739]. If the CA does not support unauthenticated GET of STAR certificates, the IdO **MUST NOT** forward the Order request. Instead, it **MUST** move the Order status to `invalid` and set the `allow-certificate-get` in the `auto-renewal` object to `false`. The same occurs in case the Order request is forwarded and the CA does not reflect the `allow-certificate-get` setting in its Order resource. The combination of `invalid` status and denied `allow-certificate-get` in the Order resource at the IdO provides an unambiguous (asynchronous) signal to the NDC about the failure reason.

#### 2.3.2.1. CNAME Installation

If one of the objects in the `identifiers` list is of type `dns`, the IdO can add the CNAME records specified in the `delegation` object to its zone, for example:

```
abc.ido.example. CNAME abc.ndc.example.
```

#### 2.3.3. Order Object Transmitted from NDC to IdO and to ACME Server (Non-STAR)

If the delegation is for a non-STAR certificate, the request object created by the NDC:

- **MUST** have a `delegation` attribute indicating the preconfigured delegation that applies to this Order;
- **MUST** have entries in the `identifiers` field for each delegated name present in the configuration; and
- **MUST** have the `allow-certificate-get` attribute set to true.

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

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://acme.ido.example/acme/acct/evOfKhNU60wg",
    "nonce": "IYBkoQfaCS80UcCn9qH8Gt",
    "url": "https://acme.ido.example/acme/new-order"
  }),
  "payload": base64url({
    "identifiers": [
      {
        "type": "dns",
        "value": "abc.ido.example"
      }
    ],
    "delegation":
      "https://acme.ido.example/acme/delegation/gm0wfLYHBen",
    "allow-certificate-get": true
  }),
  "signature": "j9JBUvMigi4zodud...acYkEKaa8gqWyZ6H"
}
```

*Figure 7: New Non-STAR Order from NDC*

The order object that is created on the IdO:

- **MUST** start in the `ready` state;
- **MUST** contain an `authorizations` array with zero elements;
- **MUST** contain the indicated `delegation` configuration; and
- **MUST** contain the indicated `allow-certificate-get` setting.

```
{
  "status": "ready",
  "expires": "2021-05-01T00:00:00Z",

  "identifiers": [
   {
     "type": "dns",
     "value": "abc.ido.example"
   }
  ],

  "delegation":
    "https://acme.ido.example/acme/delegation/gm0wfLYHBen",

  "allow-certificate-get": true,

  "authorizations": [],

  "finalize": "https://acme.ido.example/acme/order/3ZDlhYy/finalize"
}
```

*Figure 8: Non-STAR Order Resource Created on IdO*

The Order finalization by the NDC and the subsequent validation of the CSR by the IdO proceed in the same way as for the STAR case. If the CSR is successfully validated, the order object status moves to `processing` and the twin ACME protocol instance is initiated on the IdO-CA side.

The request object created by the IdO:

- **MUST** copy the identifiers sent by the NDC;
- **MUST** strip the `delegation` attribute; and
- **MUST** copy the `allow-certificate-get` attribute.

When the identifiers' authorization has been successfully completed and the certificate has been issued by the CA, the IdO:

- **MUST** move its Order resource status to `valid` and
- **MUST** copy the `certificate` field from the Order returned by the CA into its Order resource, as well as `notBefore` and `notAfter` if these fields exist.

```
{
  "status": "valid",
  "expires": "2021-05-01T00:00:00Z",

  "identifiers": [
   {
     "type": "dns",
     "value": "abc.ido.example"
   }
  ],

  "delegation":
    "https://acme.ido.example/acme/delegation/gm0wfLYHBen",

  "allow-certificate-get": true,

  "authorizations": [],

  "finalize": "https://acme.ido.example/acme/order/3ZDlhYy/finalize",

  "certificate": "https://acme.ca.example/acme/order/YtR23SsdG9"
}
```

*Figure 9: Non-STAR Order Resource Updated on IdO*

At this point of the protocol flow, the same considerations as in Section 2.3.2.1 apply.

Before forwarding the Order request to the CA, the IdO **SHOULD** ensure that the selected CA supports unauthenticated GET by inspecting the relevant settings in the CA's directory object, as per Section 2.3.5. If the CA does not support unauthenticated GET of certificate resources, the IdO **MUST NOT** forward the Order request. Instead, it **MUST** move the Order status to `invalid` and set the `allow-certificate-get` attribute to `false`. The same occurs in case the Order request is forwarded and the CA does not reflect the `allow-certificate-get` setting in its Order resource. The combination of `invalid` status and denied `allow-certificate-get` in the Order resource at the IdO provides an unambiguous (asynchronous) signal to the NDC about the failure reason.

### 2.3.4. Capability Discovery

In order to help a client discover support for this profile, the directory object of an ACME server (typically, one deployed by the IdO) contains the following attribute in the `meta` field:

delegation-enabled (optional, boolean): Boolean flag indicating support for the profile specified in this memo. An ACME server that supports this delegation profile **MUST** include this key and **MUST** set it to true.

The IdO **MUST** declare its support for delegation using `delegation-enabled` regardless of whether it supports delegation of STAR certificates, non-STAR certificates, or both.

In order to help a client discover support for certificate fetching using unauthenticated HTTP GET, the directory object of an ACME server (typically, one deployed by the CA) contains the following attribute in the `meta` field:

allow-certificate-get (optional, boolean): See Section 2.3.5.

#### 2.3.5. Negotiating an Unauthenticated GET

In order to enable the name delegation of non-STAR certificates, this document defines a mechanism that allows a server to advertise support for accessing certificate resources via unauthenticated GET (in addition to POST-as-GET) and a client to enable this service with per-Order granularity.

It is worth pointing out that the protocol elements described in this section have the same names and semantics as those introduced in Section 3.4 of [RFC8739] for the STAR use case (except, of course, they apply to the certificate resource rather than the star-certificate resource). However, they differ in terms of their position in the directory meta and order objects; rather than being wrapped in an `auto-renewal` subobject, they are located at the top level.

A server states its availability to grant unauthenticated access to a client's Order certificate by setting the `allow-certificate-get` attribute to `true` in the `meta` field inside the directory object:

allow-certificate-get (optional, boolean): If this field is present and set to `true`, the server allows GET (and HEAD) requests to certificate URLs.

A client states its desire to access the issued certificate via unauthenticated GET by adding an `allow-certificate-get` attribute to the payload of its newOrder request and setting it to `true`.

allow-certificate-get (optional, boolean): If this field is present and set to `true`, the client requests the server to allow unauthenticated GET (and HEAD) to the certificate associated with this Order.

If the server accepts the request, it **MUST** reflect the attribute setting in the resulting order object.

Note that even when the use of unauthenticated GET has been agreed upon, the server **MUST** also allow POST-as-GET requests to the certificate resource.

#### 2.3.6. Terminating the Delegation

Identity delegation is terminated differently depending on whether or not this is a STAR certificate.

##### 2.3.6.1. By Cancellation (STAR)

The IdO can terminate the delegation of a STAR certificate by requesting its cancellation (see Section 3.1.2 of [RFC8739]).

Cancellation of the ACME STAR certificate is a prerogative of the IdO. The NDC does not own the relevant account key on the CA; therefore, it can't issue a cancellation request for the STAR certificate. Potentially, since it holds the STAR certificate's private key, it could request the revocation of a single STAR certificate. However, STAR explicitly disables the revokeCert interface.

Shortly after the automatic renewal process is stopped by the IdO, the last issued STAR certificate expires and the delegation terminates.

#### 2.3.6.2. By Revocation (Non-STAR)

The IdO can terminate the delegation of a non-STAR certificate by requesting it to be revoked using the `revokeCert` URL exposed by the CA.

According to Section 7.6 of [RFC8555], the revocation endpoint can be used with either the account key pair or the certificate key pair. In other words, an NDC that learns the `revokeCert` URL of the CA (which is publicly available via the CA's directory object) would be able to revoke the certificate using the associated private key. However, given the trust relationship between the NDC and IdO expected by the delegation trust model (Section 7.1), as well as the lack of incentives for the NDC to prematurely terminate the delegation, this does not represent a significant security risk.

## 2.4. Proxy Behavior

There are cases where the ACME Delegation flow should be proxied, such as the use case described in Section 5.1.2. This section describes the behavior of such proxies.

An entity implementing the IdO server role -- an "ACME Delegation server" -- may behave, on a per-identity case, either as a proxy into another ACME Delegation server or as an IdO and obtain a certificate directly. The determining factor is whether it can successfully be authorized by the next-hop ACME server for the identity associated with the certificate request.

The identities supported by each server and the disposition for each of them are preconfigured.

Following is the proxy's behavior for each of the messages exchanged in the ACME Delegation process:

New-order request:

- The complete `identifiers` attribute **MUST** be copied as is.
- Similarly, the `auto-renewal` object **MUST** be copied as is.

New-order response:

- The `status`, `expires`, `authorizations`, `identifiers`, and `auto-renewal` attributes/objects **MUST** be copied as is.
- The `finalize` URL is rewritten so that the `finalize` request will be made to the proxy.
- Similarly, the `Location` header **MUST** be rewritten to point to an order object on the proxy.

- Any `Link` relations **MUST** be rewritten to point to the proxy.

Get Order response:

- The `status`, `expires`, `authorizations`, `identifiers`, and `auto-renewal` attributes/objects **MUST** be copied as is.
- Similarly, the `star-certificate` URL (or the `certificate` URL in case of non-STAR requests) **MUST** be copied as is.
- The `finalize` URL is rewritten so that the `finalize` request will be made to the proxy.
- The `Location` header **MUST** be rewritten to point to an order object on the proxy.
- Any `Link` relations **MUST** be rewritten to point to the proxy.

`finalize` request:

- The CSR **MUST** be copied as is.

`finalize` response:

- The `Location` header, `Link` relations, and the `finalize` URLs are rewritten as for Get Order.

We note that all the above messages are authenticated; therefore, each proxy must be able to authenticate any subordinate server.

# 3. CA Behavior

Although most of this document, and in particular Section 2, is focused on the protocol between the NDC and IdO, the protocol does affect the ACME server running in the CA. A CA that wishes to support certificate delegation **MUST** also support unauthenticated certificate fetching, which it declares using `allow-certificate-get` (Section 2.3.5, Paragraph 3).

# 4. CSR Template

The CSR template is used to express and constrain the shape of the CSR that the NDC uses to request the certificate. The CSR is used for every certificate created under the same delegation. Its validation by the IdO is a critical element in the security of the whole delegation mechanism.

Instead of defining every possible CSR attribute, this document takes a minimalist approach by declaring only the minimum attribute set and deferring the registration of further, more-specific attributes to future documents.

## 4.1. Template Syntax

The template is a JSON document. Each field (with the exception of `keyTypes`, see below) denotes one of the following:

- A mandatory field where the template specifies the literal value of that field. This is denoted by a literal string, such as `abc.ido.example`.

- A mandatory field where the content of the field is defined by the client. This is denoted by `**`.
- An optional field where the client decides whether the field is included in the CSR and, if so, what its value is. This is denoted by `*`.

The NDC **MUST NOT** include any fields in the CSR, including any extensions, unless they are specified in the template.

The structure of the template object is defined by the Concise Data Definition Language (CDDL) [RFC8610] document in Appendix A. An alternative, nonnormative JSON Schema syntax is given in Appendix B. While the CSR template must follow the syntax defined here, neither the IdO nor the NDC are expected to validate it at runtime.

The `subject` field and its subfields are mapped into the `subject` field of the CSR, as per Section 4.1.2.6 of [RFC5280]. Other extension fields of the CSR template are mapped into the CSR according to the table in Section 6.5.

The `subjectAltName` field is currently defined for the following identifiers: DNS names, email addresses, and URIs. New identifier types may be added in the future by documents that extend this specification. Each new identifier type **SHALL** have an associated identifier validation challenge that the CA can use to obtain proof of the requester's control over it.

The `keyTypes` property is not copied into the CSR. Instead, this property constrains the `SubjectPublicKeyInfo` field of the CSR, which **MUST** have the type/size defined by one of the array members of the `keyTypes` property.

When the IdO receives the CSR, it **MUST** verify that the CSR is consistent with the template contained in the `delegation` object referenced in the Order. The IdO **MAY** enforce additional constraints, e.g., by restricting field lengths. In this regard, note that a `subjectAltName` of type `DNS` can be specified using the wildcard notation, meaning that the NDC can be required (`**`) or offered the possibility (`*`) to define the delegated domain name by itself. If this is the case, the IdO **MUST** apply application-specific checks on top of the control rules already provided by the CSR template to ensure the requested domain name is legitimate according to its local policy.

## 4.2. Example

The CSR template in Figure 10 represents one possible CSR template governing the delegation exchanges provided in the rest of this document.

```
{
  "keyTypes": [
    {
      "PublicKeyType": "rsaEncryption",
      "PublicKeyLength": 2048,
      "SignatureType": "sha256WithRSAEncryption"
    },
    {
      "PublicKeyType": "id-ecPublicKey",
      "namedCurve": "secp256r1",
      "SignatureType": "ecdsa-with-SHA256"
    }
  ],
  "subject": {
    "country": "CA",
    "stateOrProvince": "**",
    "locality": "**"
  },
  "extensions": {
    "subjectAltName": {
      "DNS": [
        "abc.ido.example"
      ]
    },
    "keyUsage": [
      "digitalSignature"
    ],
    "extendedKeyUsage": [
      "serverAuth",
      "clientAuth"
    ]
  }
}
```

*Figure 10: Example CSR Template*

# 5. Further Use Cases

This nonnormative section describes additional use cases implementing the STAR certificate delegation in nontrivial ways.

## 5.1. CDN Interconnection (CDNI)

[HTTPS-DELEGATION] discusses several solutions addressing different delegation requirements for the CDN Interconnection (CDNI) environment. This section discusses two of the stated requirements in the context of the STAR delegation workflow.

This section uses specific CDNI terminology, e.g., Upstream CDN (uCDN) and Downstream (dCDN), as defined in [RFC7336].

#### 5.1.1. Multiple Parallel Delegates

In some cases, the content owner (IdO) would like to delegate authority over a website to multiple NDCs (CDNs). This could happen if the IdO has agreements in place with different regional CDNs for different geographical regions or if a "backup" CDN is used to handle overflow traffic by temporarily altering some of the CNAME mappings in place. The STAR delegation flow enables this use case naturally, since each CDN can authenticate separately to the IdO (via its own separate account) specifying its CSR, and the IdO is free to allow or deny each certificate request according to its own policy.

#### 5.1.2. Chained Delegation

In other cases, a content owner (IdO) delegates some domains to a large CDN (uCDN), which in turn delegates to a smaller regional CDN (dCDN). The IdO has a contractual relationship with uCDN, and uCDN has a similar relationship with dCDN. However, IdO may not even know about dCDN.

If needed, the STAR protocol can be chained to support this use case: uCDN could forward requests from dCDN to IdO and forward responses back to dCDN. Whether such proxying is allowed is governed by policy and contracts between the parties.

A mechanism is necessary at the interface between uCDN and dCDN, by which the uCDN can advertise:

- the names that the dCDN is allowed to use and
- the policy for creating the key material (allowed algorithms, minimum key lengths, key usage, etc.) that the dCDN needs to satisfy.

Note that such mechanism is provided by the CSR template.

##### 5.1.2.1. Two-Level Delegation in CDNI

A User Agent (UA), e.g., a browser or set-top box, wants to fetch the video resource at the following URI: `https://video.cp.example/movie`. Redirection between the content provider (CP) and upstream and downstream CDNs is arranged as a CNAME-based aliasing chain, as illustrated in Figure 11.

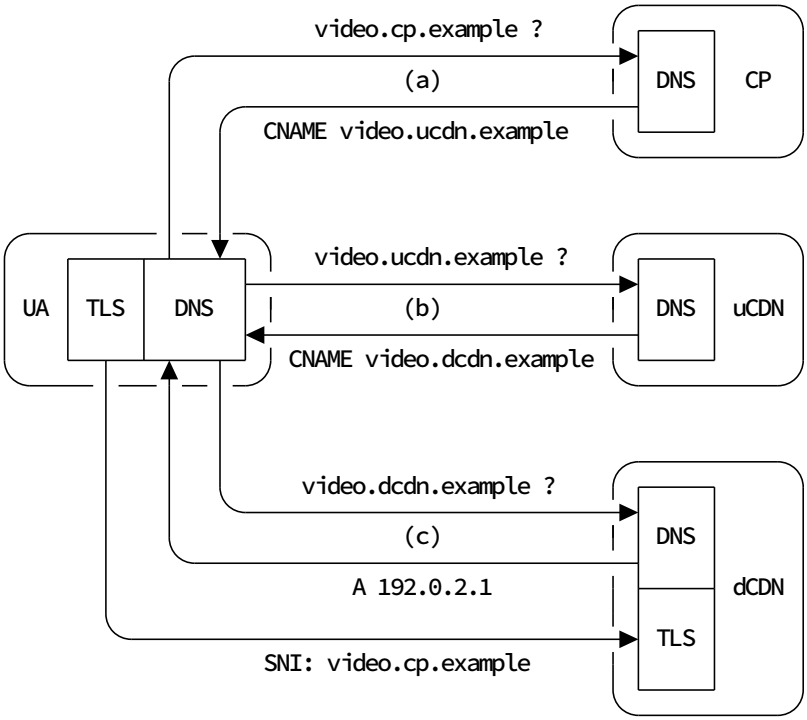


*Figure 11: DNS Redirection*

Unlike HTTP-based redirection, where the original URL is supplanted by the one found in the `Location` header of the 302 response, DNS redirection is completely transparent to the User Agent. As a result, the TLS connection to the dCDN edge is done with a Server Name Indication (SNI) equal to the `host` in the original URI -- in the example, `video.cp.example`. So, in order to successfully complete the handshake, the landing dCDN node has to be configured with a certificate whose `subjectAltName` field matches `video.cp.example`, i.e., a content provider's name.

Figure 12 illustrates the cascaded delegation flow that allows dCDN to obtain a STAR certificate that bears a name belonging to the content provider with a private key that is only known to the dCDN.

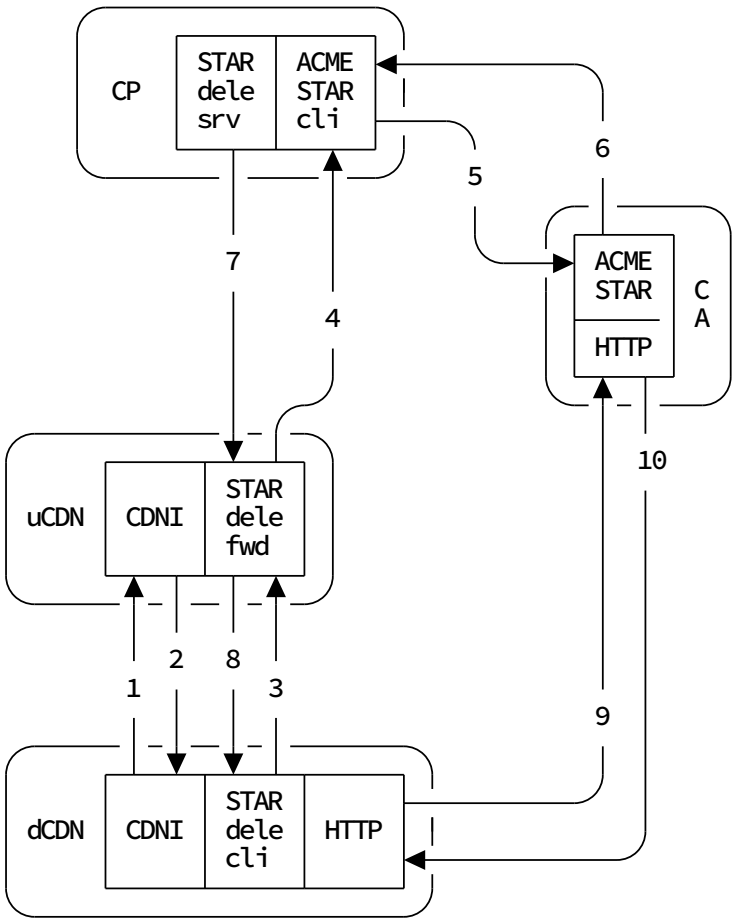


*Figure 12: Two-Level Delegation in CDNI*

uCDN is configured to delegate to dCDN, and CP is configured to delegate to uCDN, both as defined in Section 2.3.1.

1. dCDN requests CDNI path metadata to uCDN.
2. uCDN replies with, among other CDNI metadata, the STAR delegation configuration, which includes the delegated content provider's name.
3. dCDN creates a key pair and the CSR with the delegated name. It then places an order for the delegated name to uCDN.
4. uCDN forwards the received order to the content provider (CP).
5. CP creates an order for a STAR certificate and sends it to the CA. The order also requests unauthenticated access to the certificate resource.
6. After all authorizations complete successfully, the STAR certificate is issued.
7. CP notifies uCDN that the STAR certificate is available at the order's `star-certificate` URL.
8. uCDN forwards the information to dCDN. At this point, the ACME signaling is complete.

9. dCDN requests the STAR certificate using unauthenticated GET from the CA.
10. The CA returns the certificate. Now dCDN is fully configured to handle HTTPS traffic in lieu of the content provider.

Note that 9 and 10 repeat until the delegation expires or is terminated.

## 5.2. Secure Telephone Identity Revisited (STIR)

As a second use case, we consider the delegation of credentials in the STIR ecosystem [RFC9060].

This section uses STIR terminology. The term Personal Assertion Token (PASSporT) is defined in [RFC8225], and "TNAuthList" is defined in [RFC8226].

In the STIR delegated mode, a service provider SP2 -- the NDC -- needs to sign PASSporTs [RFC8225] for telephone numbers (e.g., TN=+123) belonging to another service provider, SP1 -- the IdO. In order to do that, SP2 needs a STIR certificate and a private key that includes TN=+123 in the TNAuthList [RFC8226] certificate extension.

In detail (Figure 13):

1. SP1 and SP2 agree on the configuration of the delegation -- in particular, the CSR template that applies.
2. SP2 generates a private/public key pair and sends a CSR to SP1, requesting creation of a certificate with an SP1 name, an SP2 public key, and a TNAuthList extension with the list of TNs that SP1 delegates to SP2. (Note that the CSR sent by SP2 to SP1 needs to be validated against the CSR template agreed upon in step 1.).
3. SP1 sends an order for the CSR to the CA. The order also requests unauthenticated access to the certificate resource.
4. Subsequently, after the required TNAuthList authorizations are successfully completed, the CA moves the order to a "valid" state; at the same time, the star-certificate endpoint is populated.
5. The contents of the order are forwarded from SP1 to SP2 by means of the paired "delegation" order.
6. SP2 dereferences the `star-certificate` URL in the order to fetch the rolling STAR certificate bearing the delegated identifiers.
7. The STAR certificate is returned to SP2.

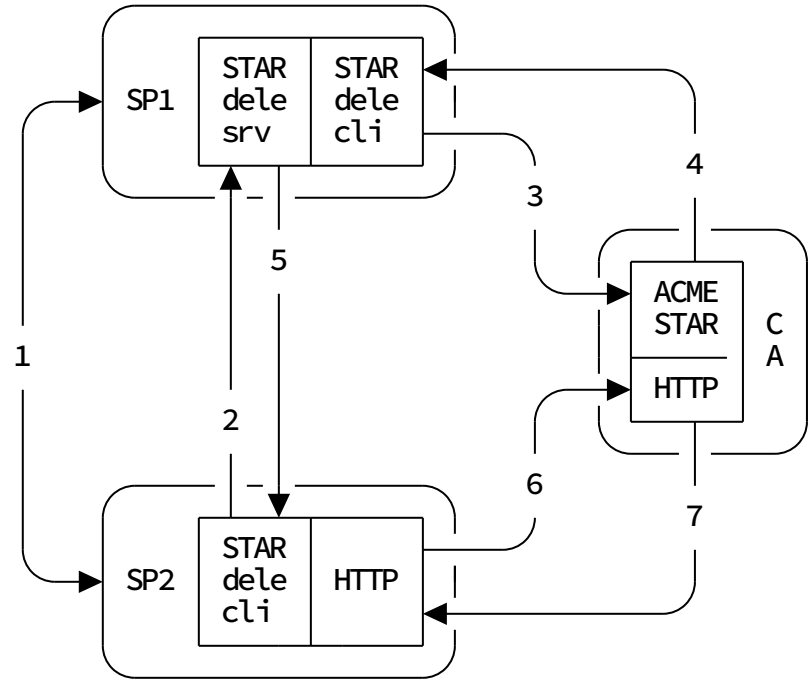


*Figure 13: Delegation in STIR*

As shown, the STAR delegation profile described in this document applies straightforwardly; the only extra requirement being the ability to instruct the NDC about the allowed TNAuthList values. This can be achieved by a simple extension to the CSR template.

# 6. IANA Considerations

## 6.1. New Fields in the "meta" Object within a Directory Object

This document adds the following entries to the "ACME Directory Metadata Fields" registry:

| Field Name | Field Type | Reference |
|---|---|---|
| delegation-enabled | boolean | RFC 9115 |
| allow-certificate-get | boolean | RFC 9115 |

*Table 1*

### 6.2. New Fields in the Order Object

This document adds the following entries to the "ACME Order Object Fields" registry:

| Field Name | Field Type | Configurable | Reference |
|---|---|---|---|
| allow-certificate-get | boolean | true | RFC 9115 |
| delegation | string | true | RFC 9115 |

*Table 2*

### 6.3. New Fields in the Account Object

This document adds the following entries to the "ACME Account Object Fields" registry:

| Field Name | Field Type | Requests | Reference |
|---|---|---|---|
| delegations | string | none | RFC 9115 |

*Table 3*

Note that the `delegations` field is only reported by ACME servers that have `delegation-enabled` set to true in their meta Object.

### 6.4. New Error Types

This document adds the following entries to the "ACME Error Types" registry:

| Type | Description | Reference |
|---|---|---|
| unknownDelegation | An unknown configuration is listed in the `delegation` attribute of the order request | RFC 9115 |

*Table 4*

### 6.5. CSR Template Extensions

IANA is requested to establish a registry, "STAR Delegation CSR Template Extensions", with "Specification Required" as its registration procedure.

Each extension registered must specify:

- an extension name,
- an extension syntax, as a reference to a CDDL document that defines this extension, and
- the extension's mapping into an X.509 certificate extension.

The initial contents of this registry are the extensions defined by the CDDL in Appendix A.

| Extension Name | Extension Syntax | Mapping to X.509 Certificate Extension |
|---|---|---|
| keyUsage | See Appendix A | [RFC5280], Section 4.2.1.3 |
| extendedKeyUsage | See Appendix A | [RFC5280], Section 4.2.1.12 |
| subjectAltName | See Appendix A | [RFC5280], Section 4.2.1.6 (note that only specific name formats are allowed: URI, DNS name, email address) |

*Table 5*

When evaluating a request for an assignment in this registry, the designated expert should follow this guidance:

- The definition must include a full CDDL definition, which the expert will validate.
- The definition must include both positive and negative test cases.
- Additional requirements that are not captured by the CDDL definition are allowed but must be explicitly specified.

# 7. Security Considerations

## 7.1. Trust Model

The ACME trust model needs to be extended to include the trust relationship between NDC and IdO. Note that once this trust link is established, it potentially becomes recursive. Therefore, there has to be a trust relationship between each of the nodes in the delegation chain; for example, in case of cascading CDNs, this is contractually defined. Note that when using standard [RFC6125] identity verification, there are no mechanisms available to the IdO to restrict the use of the delegated name once the name has been handed over to the first NDC. It is, therefore, expected that contractual measures are in place to get some assurance that redelegation is not being performed.

## 7.2. Delegation Security Goal

Delegation introduces a new security goal: only an NDC that has been authorized by the IdO, either directly or transitively, can obtain a certificate with an IdO identity.

From a security point of view, the delegation process has five separate parts:

1. enabling a specific third party (the intended NDC) to submit requests for delegated certificates

2. making sure that any request for a delegated certificate matches the intended "shape" in terms of delegated identities as well as any other certificate metadata, e.g., key length, x.509 extensions, etc.
3. serving the certificate back to the NDC
4. handling revocation of the delegation
5. handling revocation of the certificate itself

The first part is covered by the NDC's ACME account that is administered by the IdO, whose security relies on the correct handling of the associated key pair. When a compromise of the private key is detected, the delegate **MUST** use the account deactivation procedures defined in Section 7.3.6 of [RFC8555].

The second part is covered by the act of checking an NDC's certificate request against the intended CSR template. The steps of shaping the CSR template correctly, selecting the right CSR template to check against the presented CSR, and making sure that the presented CSR matches the selected CSR template are all security relevant.

The third part builds on the trust relationship between NDC and IdO that is responsible for correctly forwarding the certificate URL from the Order returned by the CA.

The fourth part is associated with the ability of the IdO to unilaterally remove the `delegation` object associated with the revoked identity, therefore, disabling any further NDC requests for such identity. Note that, in more extreme circumstances, the IdO might decide to disable the NDC account, thus entirely blocking any further interaction.

The fifth is covered by two different mechanisms, depending on the nature of the certificate. For STAR, the IdO shall use the cancellation interface defined in Section 2.3 of [RFC8739]. For non-STAR, the certificate revocation interface defined in Section 7.6 of [RFC8555]) is used.

The ACME account associated with the delegation plays a crucial role in the overall security of the presented protocol. This, in turn, means that (in delegation scenarios) the security requirements and verification associated with an ACME account may be more stringent than in base ACME deployments, since the out-of-band configuration of delegations that an account is authorized to use (combined with account authentication) takes the place of the normal ACME authorization challenge procedures. Therefore, the IdO **MUST** ensure that each account is associated with the exact policies (via their matching `delegation` objects) that define which domain names can be delegated to the account and how. The IdO is expected to use out-of-band means to preregister each NDC to the corresponding account.

## 7.3. New ACME Channels

Using the model established in Section 10.1 of [RFC8555], we can decompose the interactions of the basic delegation workflow, as shown in Figure 14.

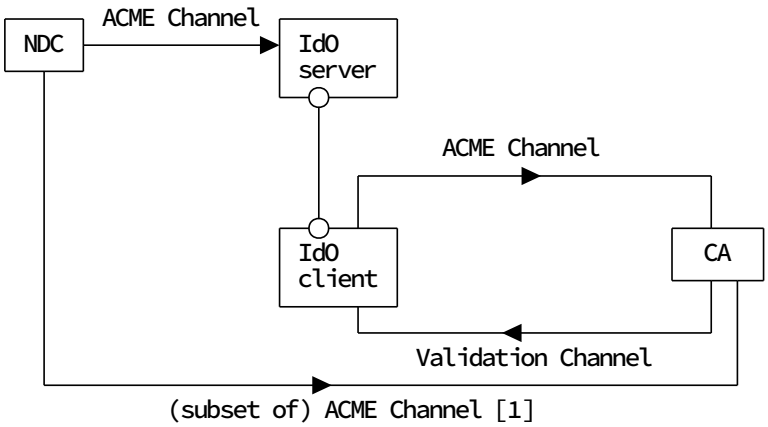


*Figure 14: Delegation Channels Topology*

The considerations regarding the security of the ACME Channel and Validation Channel discussed in [RFC8555] apply verbatim to the IdO-CA leg. The same can be said for the ACME Channel on the NDC-IdO leg. A slightly different set of considerations apply to the ACME Channel between the NDC and CA, which consists of a subset of the ACME interface comprising two API endpoints: the unauthenticated certificate retrieval and, potentially, non-STAR revocation via certificate private key. No specific security considerations apply to the former, but the privacy considerations in Section 6.3 of [RFC8739] do. With regard to the latter, it should be noted that there is currently no means for an IdO to disable authorizing revocation based on certificate private keys. So, in theory, an NDC could use the revocation API directly with the CA, therefore, bypassing the IdO. The NDC **SHOULD NOT** directly use the revocation interface exposed by the CA unless failing to do so would compromise the overall security, for example, if the certificate private key is compromised and the IdO is not currently reachable.

All other security considerations from [RFC8555] and [RFC8739] apply as is to the delegation topology.

## 7.4. Restricting CDNs to the Delegation Mechanism

When a website is delegated to a CDN, the CDN can in principle modify the website at will, e.g., create and remove pages. This means that a malicious or breached CDN can pass the ACME (as well as common non-ACME) HTTPS-based validation challenges and generate a certificate for the site. This is true regardless of whether or not the CNAME mechanisms defined in the current document is used.

In some cases, this is the desired behavior; the domain holder trusts the CDN to have full control of the cryptographic credentials for the site. However, this document assumes a scenario where the domain holder only wants to delegate restricted control and wishes to retain the capability to cancel the CDN's credentials at a short notice.

The following is a possible mitigation when the IdO wishes to ensure that a rogue CDN cannot issue unauthorized certificates:

- The domain holder makes sure that the CDN cannot modify the DNS records for the domain. The domain holder should ensure it is the only entity authorized to modify the DNS zone. Typically, it establishes a CNAME resource record from a subdomain into a CDN-managed domain.
- The domain holder uses a Certification Authority Authorization (CAA) record [RFC8659] to restrict certificate issuance for the domain to specific CAs that comply with ACME and are known to implement [RFC8657].
- The domain holder uses the ACME-specific CAA mechanism [RFC8657] to restrict issuance to a specific CA account that is controlled by it and **MUST** require "dns-01" as the sole validation method.

We note that the above solution may need to be tweaked depending on the exact capabilities and authorization flows supported by the selected CA. In addition, this mitigation may be bypassed if a malicious or misconfigured CA does not comply with CAA restrictions.

# 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>.

**[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>.

**[RFC7807]** Nottingham, M. and E. Wilde, "Problem Details for HTTP APIs", RFC 7807, DOI 10.17487/RFC7807, March 2016, <https://www.rfc-editor.org/info/rfc7807>.

**[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>.

**[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>.

**[RFC8610]** Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/info/rfc8610>.

**[RFC8739]** Sheffer, Y., Lopez, D., Gonzalez de Dios, O., Pastor Perales, A., and T. Fossati, "Support for Short-Term, Automatically Renewed (STAR) Certificates in the Automated Certificate Management Environment (ACME)", RFC 8739, DOI 10.17487/RFC8739, March 2020, <https://www.rfc-editor.org/info/rfc8739>.

## 8.2. Informative References

**[HTTPS-DELEGATION]** Fieau, F., Stephan, E., and S. Mishra, "CDNI extensions for HTTPS delegation", Work in Progress, Internet-Draft, draft-ietf-cdni-interfaces-https-delegation-05, 12 March 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-cdni-interfaces-https-delegation-05>.

**[json-schema-07]** Wright, A., Andrews, H., and B. Hutton, "JSON Schema Validation: A Vocabulary for Structural Validation of JSON", Work in Progress, Internet-Draft, draft-handrews-json-schema-validation-02, 17 September 2019, <https://datatracker.ietf.org/doc/html/draft-handrews-json-schema-validation-02>.

**[MGLT-LURK-TLS13]** Migault, D., "LURK Extension version 1 for (D)TLS 1.3 Authentication", Work in Progress, Internet-Draft, draft-mglt-lurk-tls13-05, 26 July 2021, <https://datatracker.ietf.org/doc/html/draft-mglt-lurk-tls13-05>.

**[RFC6125]** Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

**[RFC7336]** Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <https://www.rfc-editor.org/info/rfc7336>.

**[RFC8225]** Wendt, C. and J. Peterson, "PASSporT: Personal Assertion Token", RFC 8225, DOI 10.17487/RFC8225, February 2018, <https://www.rfc-editor.org/info/rfc8225>.

**[RFC8226]** Peterson, J. and S. Turner, "Secure Telephone Identity Credentials: Certificates", RFC 8226, DOI 10.17487/RFC8226, February 2018, <https://www.rfc-editor.org/info/rfc8226>.

**[RFC8657]** Landau, H., "Certification Authority Authorization (CAA) Record Extensions for Account URI and Automatic Certificate Management Environment (ACME) Method Binding", RFC 8657, DOI 10.17487/RFC8657, November 2019, <https://www.rfc-editor.org/info/rfc8657>.

**[RFC8659]** Hallam-Baker, P., Stradling, R., and J. Hoffman-Andrews, "DNS Certification Authority Authorization (CAA) Resource Record", RFC 8659, DOI 10.17487/RFC8659, November 2019, <https://www.rfc-editor.org/info/rfc8659>.

**[RFC9060]** Peterson, J., "Secure Telephone Identity Revisited (STIR) Certificate Delegation", RFC 9060, DOI 10.17487/RFC9060, August 2021, <https://www.rfc-editor.org/info/rfc9060>.

**[TLS-SUBCERTS]** Barnes, R., Iyengar, S., Sullivan, N., and E. Rescorla, "Delegated Credentials for TLS", Work in Progress, Internet-Draft, draft-ietf-tls-subcerts-10, 24 January 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-subcerts-10>.

**[TOKEN-TNAUTHLIST]** Wendt, C., Hancock, D., Barnes, M., and J. Peterson, "TNAuthList profile of ACME Authority Token", Work in Progress, Internet-Draft, draft-ietf-acme-authority-token-tnauthlist-08, 27 March 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-acme-authority-token-tnauthlist-08>.

# Appendix A. CSR Template: CDDL

Following is the normative definition of the CSR template using CDDL [RFC8610]. The CSR template **MUST** be a valid JSON document that is compliant with the syntax defined here.

There are additional constraints not expressed in CDDL that **MUST** be validated by the recipient, including:

- the value of each `subjectAltName` entry is compatible with its type and

- the parameters in each `keyTypes` entry form an acceptable combination.

```
csr-template-schema = {
  keyTypes: [ + $keyType ]
  ? subject: non-empty<distinguishedName>
  extensions: extensions
}

non-empty<M> = (M) .and ({ + any => any })

mandatory-wildcard = "**"
optional-wildcard = "*"
wildcard = mandatory-wildcard / optional-wildcard

; regtext matches all text strings but "*" and "**"
regtext = text .regexp "([^\*].*)|([\*][^\*].*)|([\*][\*].+)"

regtext-or-wildcard = regtext / wildcard

distinguishedName = {
  ? country: regtext-or-wildcard
  ? stateOrProvince: regtext-or-wildcard
  ? locality: regtext-or-wildcard
  ? organization: regtext-or-wildcard
  ? organizationalUnit: regtext-or-wildcard
  ? emailAddress: regtext-or-wildcard
  ? commonName: regtext-or-wildcard
}

$keyType /= rsaKeyType
$keyType /= ecdsaKeyType

rsaKeyType = {
  PublicKeyType: "rsaEncryption" ; OID: 1.2.840.113549.1.1.1
  PublicKeyLength: rsaKeySize
  SignatureType: $rsaSignatureType
}

rsaKeySize = uint

; RSASSA-PKCS1-v1_5 with SHA-256
$rsaSignatureType /= "sha256WithRSAEncryption"
; RSASSA-PCKS1-v1_5 with SHA-384
$rsaSignatureType /= "sha384WithRSAEncryption"
; RSASSA-PCKS1-v1_5 with SHA-512
$rsaSignatureType /= "sha512WithRSAEncryption"
; RSASSA-PSS with SHA-256, MGF-1 with SHA-256, and a 32 byte salt
$rsaSignatureType /= "sha256WithRSAandMGF1"
; RSASSA-PSS with SHA-384, MGF-1 with SHA-384, and a 48 byte salt
$rsaSignatureType /= "sha384WithRSAandMGF1"
; RSASSA-PSS with SHA-512, MGF-1 with SHA-512, and a 64 byte salt
$rsaSignatureType /= "sha512WithRSAandMGF1"

ecdsaKeyType = {
  PublicKeyType: "id-ecPublicKey" ; OID: 1.2.840.10045.2.1
  namedCurve: $ecdsaCurve
  SignatureType: $ecdsaSignatureType
}
```

```
$ecdsaCurve /= "secp256r1" ; OID: 1.2.840.10045.3.1.7
$ecdsaCurve /= "secp384r1" ; OID: 1.3.132.0.34
$ecdsaCurve /= "secp521r1" ; OID: 1.3.132.0.3

$ecdsaSignatureType /= "ecdsa-with-SHA256" ; paired with secp256r1
$ecdsaSignatureType /= "ecdsa-with-SHA384" ; paired with secp384r1
$ecdsaSignatureType /= "ecdsa-with-SHA512" ; paired with secp521r1

subjectaltname = {
  ? DNS: [ + regtext-or-wildcard ]
  ? Email: [ + regtext ]
  ? URI: [ + regtext ]
  * $$subjectaltname-extension
}

extensions = {
  ? keyUsage: [ + keyUsageType ]
  ? extendedKeyUsage: [ + extendedKeyUsageType ]
  subjectAltName: non-empty<subjectaltname>
}

keyUsageType /= "digitalSignature"
keyUsageType /= "nonRepudiation"
keyUsageType /= "keyEncipherment"
keyUsageType /= "dataEncipherment"
keyUsageType /= "keyAgreement"
keyUsageType /= "keyCertSign"
keyUsageType /= "cRLSign"
keyUsageType /= "encipherOnly"
keyUsageType /= "decipherOnly"

extendedKeyUsageType /= "serverAuth"
extendedKeyUsageType /= "clientAuth"
extendedKeyUsageType /= "codeSigning"
extendedKeyUsageType /= "emailProtection"
extendedKeyUsageType /= "timeStamping"
extendedKeyUsageType /= "OCSPSigning"
extendedKeyUsageType /= oid

oid = text .regexp "([0-2])((\.0)|(\.[1-9][0-9]*))*"
```

# Appendix B. CSR Template: JSON Schema

This appendix includes an alternative, nonnormative JSON Schema definition of the CSR template. The syntax used is that of draft 7 of JSON Schema, which is documented in [json-schema-07]. Note that later versions of this (now-expired) draft describe later versions of the JSON Schema syntax. At the time of writing, a stable reference for this syntax is not yet available, and we have chosen to use the draft version, which is currently best supported by tool implementations.

The same considerations about additional constraints checking discussed in Appendix A apply here as well.

```
{
  "title": "JSON Schema for the STAR Delegation CSR template",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "http://ietf.org/acme/drafts/star-delegation/csr-template",
  "$defs": {
    "distinguished-name": {
      "$id": "#distinguished-name",
      "type": "object",
      "minProperties": 1,
      "properties": {
        "country": {
          "type": "string"
        },
        "stateOrProvince": {
          "type": "string"
        },
        "locality": {
          "type": "string"
        },
        "organization": {
          "type": "string"
        },
        "organizationalUnit": {
          "type": "string"
        },
        "emailAddress": {
          "type": "string"
        },
        "commonName": {
          "type": "string"
        }
      },
      "additionalProperties": false
    },
    "rsaKeyType": {
      "$id": "#rsaKeyType",
      "type": "object",
      "properties": {
        "PublicKeyType": {
          "type": "string",
          "const": "rsaEncryption"
        },
        "PublicKeyLength": {
          "type": "integer"
        },
        "SignatureType": {
          "type": "string",
          "enum": [
            "sha256WithRSAEncryption",
            "sha384WithRSAEncryption",
            "sha512WithRSAEncryption",
            "sha256WithRSAandMGF1",
            "sha384WithRSAandMGF1",
            "sha512WithRSAandMGF1"
          ]
        }
      },
```

```
      "required": [
        "PublicKeyType",
        "PublicKeyLength",
        "SignatureType"
      ],
      "additionalProperties": false
    },
    "ecdsaKeyType": {
      "$id": "#ecdsaKeyType",
      "type": "object",
      "properties": {
        "PublicKeyType": {
          "type": "string",
          "const": "id-ecPublicKey"
        },
        "namedCurve": {
          "type": "string",
          "enum": [
            "secp256r1",
            "secp384r1",
            "secp521r1"
          ]
        },
        "SignatureType": {
          "type": "string",
          "enum": [
            "ecdsa-with-SHA256",
            "ecdsa-with-SHA384",
            "ecdsa-with-SHA512"
          ]
        }
      },
      "required": [
        "PublicKeyType",
        "namedCurve",
        "SignatureType"
      ],
      "additionalProperties": false
    }
  },
  "type": "object",
  "properties": {
    "keyTypes": {
      "type": "array",
      "minItems": 1,
      "items": {
        "anyOf": [
          {
            "$ref": "#rsaKeyType"
          },
          {
            "$ref": "#ecdsaKeyType"
          }
        ]
      }
    },
    "subject": {
      "$ref": "#distinguished-name"
```

```
    },
    "extensions": {
      "type": "object",
      "properties": {
        "keyUsage": {
          "type": "array",
          "minItems": 1,
          "items": {
            "type": "string",
            "enum": [
              "digitalSignature",
              "nonRepudiation",
              "keyEncipherment",
              "dataEncipherment",
              "keyAgreement",
              "keyCertSign",
              "cRLSign",
              "encipherOnly",
              "decipherOnly"
            ]
          }
        },
        "extendedKeyUsage": {
          "type": "array",
          "minItems": 1,
          "items": {
            "anyOf": [
              {
                "type": "string",
                "enum": [
                  "serverAuth",
                  "clientAuth",
                  "codeSigning",
                  "emailProtection",
                  "timeStamping",
                  "OCSPSigning"
                ]
              },
              {
                "type": "string",
                "pattern": "^([0-2])((\\.0)|(\\.[1-9][0-9]*))*$",
                "description": "Used for OID values"
              }
            ]
          }
        },
        "subjectAltName": {
          "type": "object",
          "minProperties": 1,
          "properties": {
            "DNS": {
              "type": "array",
              "minItems": 1,
              "items": {
                "anyOf": [
                  {
                    "type": "string",
                    "enum": [
```

```
                    "*",
                    "**"
                  ]
                },
                {
                  "type": "string",
                  "format": "hostname"
                }
              ]
            }
          },
          "Email": {
            "type": "array",
            "minItems": 1,
            "items": {
              "type": "string",
              "format": "email"
            }
          },
          "URI": {
            "type": "array",
            "minItems": 1,
            "items": {
              "type": "string",
              "format": "uri"
            }
          }
        },
        "additionalProperties": false
      }
    },
    "required": [
      "subjectAltName"
    ],
    "additionalProperties": false
  }
 },
 "required": [
   "extensions",
   "keyTypes"
 ],
 "additionalProperties": false
}
```

## Acknowledgements

We would like to thank the following people who contributed significantly to this document with their review comments and design proposals: Richard Barnes, Carsten Bormann, Roman Danyliw, Lars Eggert, Frédéric Fieau, Russ Housley, Ben Kaduk, Eric Kline, Sanjay Mishra, Francesca Palombini, Jon Peterson, Ryan Sleevi, Emile Stephan, and Éric Vyncke.

This work is partially supported by the European Commission under Horizon 2020 grant agreement no. 688421 Measurement and Architecture for a Middleboxed Internet (MAMI). This support does not imply endorsement.

## Authors' Addresses

**Yaron Sheffer**
Intuit
Email: yaronf.ietf@gmail.com

**Diego López**
Telefonica I+D
Email: diego.r.lopez@telefonica.com

**Antonio Agustín Pastor Perales**
Telefonica I+D
Email: antonio.pastorperales@telefonica.com

**Thomas Fossati**
ARM
Email: thomas.fossati@arm.com