Automated Certificate Management Environment (ACME) Renewal Information (ARI) Extension

Abstract

This document specifies how an ACME server may provide suggestions to ACME clients as to when they should attempt to renew their certificates. This allows servers to mitigate load spikes, and ensures clients do not make false assumptions about appropriate certificate renewal periods.

Current Implementations

Draft note: this section will be removed by the editor before final publication.

Let's Encrypt's Boulder software fully implements the server side of an earlier version of this draft, and that implementation is deployed in both the Production and Staging environments. Google Trust Services has done the same. The Pebble ACME server intended for testing client ACME implementations also supports ARI.

Client implementations include Lego, eggsampler, Caddy, CertMagic, ACMEz, win-acme, Tailscale, Certify the Web, acme4j, and Ansible's community.crypto.

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 2 December 2024.
1. Introduction

Most ACME [RFC8555] clients today choose when to attempt to renew a certificate in one of three ways. They may be configured to renew at a specific interval (e.g. via cron); they may parse the issued certificate to determine its expiration date and renew a specific amount of time before then; or they may parse the issued certificate and renew when some percentage of its validity period has passed. The first two techniques create significant barriers against the issuing Certification Authority (CA) changing certificate lifetimes. All three techniques lead to load clustering for the issuing CA.
Allowing issuing CAs to suggest a period in which clients should renew their certificates enables for dynamic time-based load balancing. This allows a CA to better respond to exceptional circumstances. For example, a CA could suggest that clients renew prior to a mass-revocation event to mitigate the impact of the revocation, or a CA could suggest that clients renew earlier than they normally would to reduce the size of an upcoming mass-renewal spike.

This document specifies a mechanism by which ACME servers may provide suggested renewal windows to ACME clients, and by which ACME clients may inform ACME servers that a certificate has been renewed and replaced.

2. 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.

Throughout this document, the word "renewal" and its variants are taken to encompass any combination of "Renewal", "Re-Key", and "Modification" as defined in [RFC3647].

This document assumes that the certificates being issued by the ACME server are in compliance with [RFC5280], and in particular contain the Authority Key Identifier extension and the keyIdentifier field within that extension.

3. Extensions to the Directory Object

An ACME server which wishes to provide renewal information MUST include a new field, renewalInfo, in its directory object.

<table>
<thead>
<tr>
<th>Field</th>
<th>URL in Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>renewalInfo</td>
<td>Renewal info</td>
</tr>
</tbody>
</table>

Table 1
4. Getting Renewal Information

4.1. The "renewalInfo" Resource

The "renewalInfo" resource is a new resource type introduced to ACME protocol. This new resource allows clients to query the server for suggestions on when they should renew certificates.

To request the suggested renewal information for a certificate, the client sends a GET request to a path under the server's renewalInfo URL.

The path component is a unique identifier for the certificate in question. The unique identifier is constructed by concatenating the base64url-encoding [RFC4648] of the bytes of the keyIdentifier field of certificate's Authority Key Identifier (AKI) [RFC5280] extension, a literal period, and the base64url-encoding of the bytes of the DER encoding of the certificate's Serial Number (without the tag and length bytes). All trailing "=" characters MUST be stripped from both parts of the unique identifier.

Thus the full request url is constructed as follows, where the "||" operator indicates string concatenation and the renewalInfo url is taken from the Directory object:

```url = renewalInfo || '/' || base64url(AKI) || '.' || base64url(Serial)```
For example, to request renewal information for the end-entity certificate given in Appendix A, the client would make the request as follows:

1. The keyIdentifier field of the certificate's AKI extension has the hexadecimal bytes 69:88:5B:6B:87:46:40:41:E1:B3:7B:84:7B:A0:AE:2C:DE:01:C8:D4 as its ASN.1 Octet String value. The base64url encoding of those bytes is aYhba4dGQEHhs3uEe6CuLN4ByNQ=.
2. The certificate's Serial Number field has the hexadecimal bytes 00:87:65:43:21 as its DER encoding (note the leading zero byte to ensure the serial number remains positive despite the leading 1 bit in 0x87). The base64url encoding of those bytes is AIdlQyE=.
3. Stripping the trailing padding characters and concatenating with the separator, the unique identifier is therefore aYhba4dGQEHhs3uEe6CuLN4ByNQ.AIdlQyE, and the client makes the request (split onto multiple lines for readability):

```
GET https://example.com/acme/renewal-info/
aYhba4dGQEHhs3uEe6CuLN4ByNQ.AIdlQyE
```

4.2. RenewalInfo Objects

The structure of an ACME renewalInfo resource is as follows:

```
suggestedWindow (object, required): A JSON object with two keys, "start" and "end", whose values are timestamps, encoded in the format specified in [RFC3339], which bound the window of time in which the CA recommends renewing the certificate.

explanationURL (string, optional): A URL pointing to a page which may explain why the suggested renewal window is what it is. For example, it may be a page explaining the CA's dynamic load-balancing strategy, or a page documenting which certificates are affected by a mass revocation event. Conforming clients **SHOULD** provide this URL to their operator, if present.
```
The server **SHOULD** include a Retry-After header indicating the polling interval that the ACME server recommends. Conforming clients **SHOULD** query the renewalInfo URL again after the Retry-After period has passed, as the server may provide a different suggestedWindow.

Conforming clients **MUST** attempt renewal at a time of their choosing based on the suggested renewal window. The following algorithm is **RECOMMENDED** for choosing a renewal time:

1. Query the renewalInfo resource to get a suggested renewal window.
2. Select a uniform random time within the suggested window.
3. If the selected time is in the past, attempt renewal immediately.
4. Otherwise, if the client can schedule itself to attempt renewal at exactly the selected time, do so.
5. Otherwise, if the selected time is before the next time that the client would wake up normally, attempt renewal immediately.
6. Otherwise, sleep until the Retry-After period has passed, or until the next normal wake time, and return to Step 1.

In all cases, renewal attempts are subject to the client's existing error backoff and retry intervals.

In particular, cron-based clients may find they need to increase their run frequency to check ARI more frequently. Those clients **SHOULD** need to store information about failures so that increasing their run frequency doesn't lead to retrying failures without proper backoff. Typical information stored should include: number of failures for a given order (defined by the set of names on the order), and time of the most recent failure.

If the client receives no response or a malformed response (e.g. an end timestamp which is equal to or precedes the start timestamp), it **SHOULD** make its own determination of when to renew the certificate, and **MAY** retry the renewalInfo request with appropriate exponential backoff behavior.
5. Extensions to the Order Object

In order to convey information regarding which certificate requests represent renewals of previous certificates, a new field is added to the Order object:

replaces (string, optional): A string uniquely identifying a previously-issued certificate which this order is intended to replace. This unique identifier is constructed in the same way as the path component for GET requests described above.

Clients SHOULD include this field in New Order requests if there is a clear predecessor certificate, as is the case for most certificate renewals.

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

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://example.com/acme/acct/evOfKhNU60wg",
    "nonce": "5XJ1L3LeKMG7tR6pA00clA",
    "url": "https://example.com/acme/new-order"
  }),
  "payload": base64url({
    "identifiers": [
      { "type": "dns", "value": "example.com" }
    ],
    "replaces": "aYhba4dGQEHhs3uEe6CuLN4ByNQ.AIdlQyE"
  }),
  "signature": "H6ZXtGjTZyUnPeKn...wEA4TkLdh3e454g"
}

Servers SHOULD check that the identified certificate and the New Order request correspond to the same ACME Account, that they share at least one identifier, and that the identified certificate has not already been marked as replaced by a different Order that is not "invalid". Correspondence checks beyond this (such as requiring exact identifier matching) are left up to Server policy. If any of these checks fail, the Server SHOULD reject the new-order request.

If the Server accepts a new-order request with a "replaces" field, it MUST reflect that field in the response and in subsequent requests for the corresponding Order object.
This replacement information may serve many purposes, including but not limited to:

- granting New Order requests which arrive during the suggested renewal window of their identified predecessor certificate higher priority or allow them to bypass rate limits, if the Server's policy uses such;
- tracking the replacement of certificates which have been affected by a compliance incident, so that they can be revoked immediately after they are replaced; and
- tying together certificates issued under the same contract with an entity identified by External Account Binding.

6. Security Considerations

The extensions to the ACME protocol described in this document build upon the Security Considerations and threat model defined in [RFC8555], Section 10.1.

This document specifies that renewalInfo resources **MUST** be exposed and accessed via unauthenticated GET requests, a departure from RFC8555’s requirement that clients must send POST-as-GET requests to fetch resources from the server. This is because the information contained in renewalInfo resources is not considered confidential, and because allowing renewalInfo to be easily cached is advantageous to shed load from clients which do not respect the Retry-After header.

7. IANA Considerations

7.1. ACME Resource Type

IANA will add the following entry to the "ACME Resource Types" registry within the "Automated Certificate Management Environment (ACME) Protocol" registry group at [https://www.iana.org/assignments/acme](https://www.iana.org/assignments/acme):

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Resource Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>renewalInfo</td>
<td>Renewal Info object</td>
<td>This document</td>
</tr>
</tbody>
</table>

Table 2

7.2. ACME Renewal Info Object Fields

IANA will add the following new registry to the "Automated Certificate Management Environment (ACME) Protocol" registry group at [https://www.iana.org/assignments/acme](https://www.iana.org/assignments/acme):

Registry Name: ACME Renewal Info Object Fields

Registration Procedure: Specification Required
7.3. ACME Order Object Fields

IANA will add the following entry to the "ACME Order Object Fields" registry within the "Automated Certificate Management Environment (ACME) Protocol" registry group at https://www.iana.org/assignments/acme:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Field Type</th>
<th>Configurable</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>replaces</td>
<td>string</td>
<td>true</td>
<td>This document</td>
</tr>
</tbody>
</table>

8. References

8.1. Normative References


8.2. Informative References


Appendix A. Example Certificate

-----BEGIN CERTIFICATE-----
MIIBQzCB66ADAgECAgUAh2VDDIbAKBgqhkjOPQQDAjAVMRMwEQYDVQDEwpFeGFt
cGxlENBMCY2zAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAy

Acknowledgments

My thanks to Roland Shoemaker and Jacob Hoffman-Andrews for coming up with the initial idea of ARI and for helping me learn the IETF process. Thanks also to Samantha Frank, Phil Porada, Matt Holt, Ilari Liusvaara, and Wouter Tinus for contributing client implementations, and to Freddy Zhang for contributing an independent server implementation. Finally, thanks to Rob Stradling, Andrew Ayer, and J.C. Jones for providing meaningful feedback and suggestions which significantly improved this specification.

Author's Address

A. Gable
Internet Security Research Group

Email: aaron@letsencrypt.org