Stream:	Internet Engineering Task Force (IETF)
RFC:	9773
Category:	Standards Track
Published:	May 2025
ISSN:	2070-1721
Author:	A. Gable
	Internet Security Research Group

RFC 9773
ACME Renewal Information (ARI) Extension

Abstract

This document specifies how an Automated Certificate Management Environment (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 that clients do not make false assumptions about appropriate certificate renewal periods.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Conventions and Definitions 3
3. Extensions to the Directory Object 4
4. Getting Renewal Information 4
  4.1. The RenewalInfo Resource 4
  4.2. RenewalInfo Objects 5
  4.3. Schedule for Checking the RenewalInfo Resource 6
    4.3.1. Server Choice of Retry-After 7
    4.3.2. Client Handling of Retry-After 7
    4.3.3. Error Handling 7
5. Extensions to the Order Object 7
6. Security Considerations 9
7. IANA Considerations 9
  7.1. ACME Resource Type 9
  7.2. ACME RenewalInfo Object Fields 9
  7.3. ACME Order Object Fields 10
  7.4. ACME Error Types 10
8. References 11
  8.1. Normative References 11
  8.2. Informative References 11
Appendix A. Example Certificate 12
Acknowledgments 12
Author's Address 12

1. Introduction

Most ACME [RFC8555] clients today choose when to attempt to renew a certificate in one of three ways:

1. they may be configured to renew at a specific interval (e.g., via cron),
2. they may parse the issued certificate to determine its expiration date and renew a specific amount of time before then, or
3. they may parse the issued certificate and renew when some percentage of its validity period has passed.

The first two create significant barriers against the issuing Certification Authority (CA) changing certificate lifetimes. All three ways may lead to load clustering for the issuing CA due to its inability to schedule renewal requests.

Allowing issuing CAs to suggest a period in which clients should renew their certificates enables 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 the ACME Renewal Information (ARI) extension, a mechanism by which ACME servers may provide suggested renewal windows to ACME clients and by which ACME clients may inform ACME servers that they have successfully renewed and replaced a certificate.

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 that wishes to provide renewal information **MUST** include a new field, "`renewalInfo`", in its directory object.

Field	URL in Value
renewalInfo	Renewal information

Table 1

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "newNonce": "https://acme.example.com/new-nonce",
  "newAccount": "https://acme.example.com/new-account",
  "newOrder": "https://acme.example.com/new-order",
  "newAuthz": "https://acme.example.com/new-authz",
  "revokeCert": "https://acme.example.com/revoke-cert",
  "keyChange": "https://acme.example.com/key-change",
  "renewalInfo": "https://acme.example.com/renewal-info",
  "meta": {
    "termsOfService": "https://example.com/acme/terms",
    "website": "https://example.com/acme/docs",
    "caaIdentities": ["example.com"],
    "externalAccountRequired": false
  }
}
```

4. Getting Renewal Information

4.1. The RenewalInfo Resource

The RenewalInfo resource is a new resource type introduced to the 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 an unauthenticated 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 keyIdentifier field of the certificate's Authority Key Identifier (AKI) [RFC5280] extension, the period character ".", and the base64url encoding of the DER-encoded Serial Number field (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 (split onto multiple lines for readability), where the "`||`" operator indicates string concatenation and the `renewalInfo` URL is taken from the Directory object:

```
url = renewalInfo || '/' ||
      base64url(AKI keyIdentifier) || '.' || 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:

```
GET /renewal-info/aYhba4dGQEHhs3uEe6CuLN4ByNQ.AIdlQyE HTTP/1.1
Host: acme.example.com
Accept: application/json
```

4.2. RenewalInfo Objects

The structure of an ACME RenewalInfo object 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 that may explain why the suggested renewal window has its current value. 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. Clients **SHOULD** provide this URL to their operator, if present.

```
HTTP/1.1 200 OK
Content-Type: application/json
Retry-After: 21600

{
  "suggestedWindow": {
    "start": "2025-01-02T04:00:00Z",
    "end": "2025-01-03T04:00:00Z"
  },
  "explanationURL": "https://acme.example.com/docs/ari"
}
```

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. Make a `renewalInfo` request 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 time indicated by the Retry-After header 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 will 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.

A RenewalInfo object in which the `end` timestamp equals or precedes the `start` timestamp is invalid. Servers **MUST NOT** serve such a response, and clients **MUST** treat one as though they failed to receive any response from the server (e.g., retry at an appropriate interval, renew on a fallback schedule, etc.).

4.3. Schedule for Checking the RenewalInfo Resource

Clients **SHOULD** fetch a certificate's RenewalInfo immediately after issuance.

During the lifetime of a certificate, the renewal information needs to be fetched frequently enough that clients learn about changes in the suggested window quickly, but without overwhelming the server. This protocol uses the Retry-After header [RFC9110] to indicate to clients how often to retry. Note that in other HTTP applications, Retry-After often indicates the minimum time to wait before retrying a request. In this protocol, it indicates the desired (i.e., both requested minimum and maximum) amount of time to wait.

Clients **MUST NOT** check a certificate's RenewalInfo after the certificate has expired. Clients **MUST NOT** check a certificate's RenewalInfo after they consider the certificate to be replaced (for instance, after a new certificate for the same identifiers has been received and configured).

4.3.1. Server Choice of Retry-After

Servers set the Retry-After header based on their requirements on how quickly to perform a revocation. For instance, a server that needs to revoke certificates within 24 hours of notification of a problem might choose to reserve twelve hours for investigation, six hours for clients to fetch updated RenewalInfo objects, and six hours for clients to perform a renewal. Setting a small value for Retry-After means that clients can respond more quickly but also incurs more load on the server. Servers should estimate their expected load based on the number of clients, keeping in mind that third parties may also monitor `renewalInfo` endpoints.

4.3.2. Client Handling of Retry-After

After an initial fetch of a certificate's RenewalInfo, clients **MUST** fetch it again as soon as possible after the time indicated in the Retry-After header (backoff on errors takes priority, though). Clients **MUST** set reasonable limits on their checking interval. For example, values under one minute could be treated as if they were one minute, and values over one day could be treated as if they were one day.

4.3.3. Error Handling

Temporary errors include, for instance:

- Connection timeout
- Request timeout
- 5xx HTTP errors

On receiving a temporary error, clients **SHOULD** do exponential backoff with a capped number of tries. If all tries are exhausted, clients **MUST** treat the request as a long-term error.

Examples of long-term errors include:

- Retry-After is invalid or not present
- RenewalInfo object is invalid
- DNS lookup failure
- Connection refused
- Non-5xx HTTP error

On receiving a long-term error, clients **MUST** make the next `renewalInfo` request as soon as possible after six hours have passed (or some other locally configured default).

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 that 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 newOrder requests if there is a clear predecessor certificate, as is the case for most certificate renewals. Clients **SHOULD NOT** include this field if the ACME server has not indicated that it supports this protocol by advertising the `renewalInfo` resource in its Directory.

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

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://acme.example.com/acct/evOfKhNU60wg",
    "nonce": "5XJ1L3lEkMG7tR6pA00clA",
    "url": "https://acme.example.com/new-order"
  }),
  "payload": base64url({
    "identifiers": [
      { "type": "dns", "value": "acme.example.com" }
    ],
    "replaces": "aYhba4dGQEHhs3uEe6CuLN4ByNQ.AIdlQyE"
  }),
  "signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g"
}
```

Servers **SHOULD** check that the identified certificate and the newOrder 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 newOrder request. If the server rejects the request because the identified certificate has already been marked as replaced, it **MUST** return an HTTP 409 (Conflict) with a problem document of type "alreadyReplaced" (see Section 7.4).

If the server accepts a newOrder 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 newOrder requests that arrive during the suggested renewal window of their identified predecessor certificate higher priority or allowing them to bypass rate limits, if the server's policy uses such;
- tracking the replacement of certificates that 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 Section 10.1 of [RFC8555].

This document specifies that RenewalInfo resources are exposed and accessed via unauthenticated GET requests, a departure from the requirement in RFC 8555 that clients 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 resources to be easily cached is advantageous to shed the load from clients that do not respect the Retry-After header. As always, servers should take measures to ensure that unauthenticated requests for renewal information cannot result in denial-of-service attacks. These measures might include ensuring that a cache does not include superfluous request headers or query parameters in its cache key, instituting IP-based rate limits, or other general best-practice measures.

Note that this protocol could exhibit undesired behavior in the presence of significant clock skew between the ACME client and server. For example, if a server places the suggested renewal window wholly in the past to encourage a client to renew immediately, a client with a sufficiently slow clock might nonetheless see the window as being in the future. Similarly, a server that wishes to schedule renewals very precisely may have difficulty doing so if some clients have skewed clocks (or do no implement ARI at all). Server operators should take this concern into account when setting suggested renewal windows. However, many other protocols (including TLS handshakes themselves) fall apart with sufficient clock skew, so this is not unique to this protocol.

7. IANA Considerations

7.1. ACME Resource Type

IANA has added 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>:

Field Name	Resource Type	Reference
renewalInfo	RenewalInfo object	This document

Table 2

7.2. ACME RenewalInfo Object Fields

IANA has added the following new registry to the "Automated Certificate Management Environment (ACME) Protocol" registry group at <https://www.iana.org/assignments/acme>:

Registry Name:
ACME RenewalInfo Object Fields

Registration Procedure:
Specification Required (see [RFC8126]). The designated expert should ensure that any new fields added to this registry carry useful and unique information that does not better belong elsewhere in the ACME protocol.

Template:

Field name:	The string to be used as a field name in the JSON object
Field type:	The type of value to be provided, e.g., string, boolean, array of string
Reference:	Where this field is defined

Initial contents:

Field Name	Field Type	Reference
suggestedWindow	object	This document
explanationURL	string	This document

Table 3

7.3. ACME Order Object Fields

IANA has added 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>:

Field Name	Field Type	Configurable	Reference
replaces	string	true	This document

Table 4

7.4. ACME Error Types

IANA has added the following entry to the "ACME Error Types" registry within the "Automated Certificate Management Environment (ACME) Protocol" registry group at <https://www.iana.org/assignments/acme>:

Type	Description	Reference
alreadyReplaced	The request specified a predecessor certificate that has already been marked as replaced	This document

Table 5

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>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/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/info/rfc5280>.

[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>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

8.2. Informative References

[RFC3647] Chokhani, S., Ford, W., Sabett, R., Merrill, C., and S. Wu, "Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework", RFC 3647, DOI 10.17487/RFC3647, November 2003, <https://www.rfc-editor.org/info/rfc3647>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

Appendix A. Example Certificate

```
-----BEGIN CERTIFICATE-----
MIIBQzCB66ADAgECAgUAh2VDITAKBggqhkjOPQQDAjAVMRMwEQYDVQQDEwpFeGFt
cGxlIENBMCIYDzAwMDEwMTAxMDAwMDAwWhgPMDAwMTAxMDEwMDAwMDBaMBYxFDAS
BgNVBAMTC2V4YW1wbGUuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEeBZu
7cbpAYNXZLbbh8rNIzuOoqOOtmxA1v7cRm//AwyMwWxyHz4zfwmBhcSrf47NUAFf
qzLQ2PPQxdTXREYEnKMjMCEwHwYDVR0jBBgwFoAUaYhba4dGQEHhs3uEe6CuLN4B
yNQwCgYIKoZIzj0EAwIDRwAwRAIge09+S5TZAlw5tgtiVvuERV6cT4mfutXIlwTb
+FYN/8oCIClDsqBklhB9KAelFiYt9+6FDj3z4KGVelYM5MdsO3pK
-----END CERTIFICATE-----
```

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, 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 that significantly improved this specification.

Author's Address

A. Gable
Internet Security Research Group
Email: aaron@letsencrypt.org