Automated Certificate Management Environment
Internet-Draft
Intended status: Standards Track
Expires: 25 September 2026

S. Heurich
Fastly
H. Birge-Lee
Crosslayer Labs, Inc.
M. Slaughter
Amazon Trust Services

24 March 2026

Automated Certificate Management Environment (ACME) Challenge for Persistent DNS TXT Record Validation
draft-ietf-acme-dns-persist-01

Abstract

This document specifies "dns-persist-01", a new validation method for the Automated Certificate Management Environment (ACME) protocol. This method allows a Certification Authority (CA) to verify control over a domain by confirming the presence of a persistent DNS TXT record containing CA and account identification information. This method is particularly suited for environments where traditional challenge methods are impractical, such as multi-tenant hosting platforms, enterprise DNS environments, and IoT deployments. The validation method is designed with a strong focus on security and robustness, incorporating widely adopted industry best practices for persistent domain control validation. This design aims to make it suitable for Certification Authorities operating under various policy environments, including those that align with the CA/Browser Forum Baseline Requirements.

About This Document

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

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

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

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-acme/draft-ietf-acme-dns-persist.

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 25 September 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction 4
1.1. Robustness and Alignment with Industry Best Practices . . 4
2. Conventions and Definitions 5
3. The "dns-persist-01" Challenge 6
3.1. Challenge Object 6
4. Challenge Response and Verification 7
4.1. Validation Record Format 8
4.2. Verification Procedure 10
4.3. Multiple Issuer Support 10
4.3.1. Coexistence of Records 10
4.3.2. CA Verification Process 10
4.3.3. Security and Management Considerations 11
4.3.4. Example: Authorizing Two CAs 11
4.4. Just-in-Time Validation 12
4.5. Pre-Provisioning Records 13

5. Wildcard and Subdomain Certificate Validation 14
5.1. Scope of policy=wildcard 14
6. Subdomain Certificate Validation 14
6.1. Determining Permitted Subdomains 14
6.2. Implementation Requirements 15
6.3. Example: Subdomain Validation 15
7. Security Considerations 15
7.1. Persistent Record Risks 16
7.2. Account Binding Security 16
7.2.1. Account Key Rotation 17
7.2.2. Account URI Privacy 17
7.3. Subdomain Validation Risks 17
7.4. Cross-CA Validation Reuse 18
7.5. Record Tampering and Integrity 18
7.6. Issuer Domain Name Normalization and Limits 19
7.7. DNS Security Measures 19
7.7.1. DNSSEC 19
7.7.2. Multi-Perspective Validation 19
7.8. Validation Data Reuse and TTL Handling 20
7.9. persistUntil Parameter Considerations 20
7.10. Revocation and Invalidation of Persistent Authorizations 21
8. IANA Considerations 22
8.1. ACME Validation Methods Registry 22
8.2. Underscored and Globally Scoped DNS Node Names Registry 22
9. Implementation Considerations 22
9.1. DNS Record Size Considerations 23
9.2. Domain Name Normalization Algorithm 23
9.3. CA Implementation Guidelines 24
9.3.1. Error Handling 24
9.4. Client Implementation Guidelines 25
9.5. DNS Provider Considerations 25
10. Examples 26
10.1. Basic Validation Example (FQDN Only) 26
10.2. Wildcard Validation Example 26
10.3. Validation Example with persistUntil 27
10.4. Wildcard Validation Example with persistUntil 27
11. References 28
11.1. Normative References 28
11.2. Informative References 29
Acknowledgments 29
Authors' Addresses 30

1. Introduction

The Automated Certificate Management Environment (ACME) protocol [RFC8555] defines mechanisms for automating certificate issuance and domain validation. The existing challenge methods, "http-01" and "dns-01", require real-time interaction between the ACME client and the domain's infrastructure during the validation process. While effective for many use cases, these methods present challenges in certain deployment scenarios.

Examples include:

* Edge compute and multi-tenant hosting platforms where the entity managing the DNS zone is distinct from the tenant subscribing to the certificate.

* Organizations that wish to pre-validate domains and batch issuance operations offline or at a later time.

* Environments with strict change management processes where DNS modifications require approval workflows.

* Scenarios requiring wildcard certificates where domain control is proven once and reused over an extended period.

* Internet of Things (IoT) deployments where devices may not be able to host an HTTP service or coordinate DNS updates in real-time.

This document defines a new ACME challenge type, "dns-persist-01". This method proves control over a Fully Qualified Domain Name (FQDN) by confirming the presence of a persistent DNS TXT record containing CA and account identification information.

The record format is based on the "issue-value" syntax from [RFC8659], incorporating an issuer-domain-name and a mandatory accounturi parameter [RFC8657] that uniquely identifies the applicant's account. This design provides strong binding between the domain, the CA, and the specific account requesting validation.

1.1. Robustness and Alignment with Industry Best Practices

This validation method is designed to provide a robust and persistent mechanism for domain control verification within the ACME protocol. Its technical design incorporates widely adopted security principles and best practices for domain validation, ensuring high assurance regardless of the specific CA policy environment. These principles include, but are not limited to:

1. The use of a well-defined, unique DNS label (e.g., "_validation-persist") for persistent validation records, minimizing potential conflicts.

2. Consideration of DNS TTL values when determining the effective validity period of an authorization, balancing persistence with responsiveness to DNS changes (see Section 7.8).

3. Explicit binding of the domain validation to a specific ACME account through a unique identifier, establishing clear accountability and enhancing security against unauthorized use.

Certification Authorities operating under various trust program requirements will find this technical framework suitable for their domain validation needs, as its design inherently supports robust and auditable validation practices.

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.

DNS TXT Record Persistent DCV Domain Label The label "_validation-persist" as specified in this document. This label is consistent with industry practices for persistent domain validation.

Validation Domain Name The domain name at which the validation TXT record is provisioned. It is formed by prepending the DNS TXT Record Persistent DCV Domain Label to the FQDN being validated.

This term follows the precedent set by [RFC8555], Section 8.4, which uses "validation domain name" for the analogous _acme-challenge.<FQDN> label in the dns-01 challenge. This document avoids the term "Authorization Domain Name" because the CA/Browser Forum Baseline Requirements [cabf-br] define it as the FQDN used to obtain authorization, without any label prepended.

Issuer Domain Name A domain name disclosed by the CA in Section 4.2 of the CA’s Certificate Policy and/or Certification Practices Statement to identify the CA for the purposes of this validation method.

Note: The issuer-domain-names provided in the challenge object MAY be drawn from the machine-readable caaIdentities array in the ACME server's directory object, as specified in [RFC8555], Section 7.1.1. This creates a clearer programmatic link between the server's advertised identities and the challenge object.

Validation Data Reuse Period The period during which a CA may rely on validation data, as defined by the CA's practices and applicable requirements.

persistUntil An optional parameter in the validation record that specifies the timestamp after which the validation record should no longer be considered valid by CAs. The value MUST be a base-10 encoded integer representing a UNIX timestamp (the number of seconds since 1970-01-01T00:00:00Z ignoring leap seconds).

3. The "dns-persist-01" Challenge

The "dns-persist-01" challenge allows an ACME client to demonstrate control over an FQDN by proving it can provision a DNS TXT record containing specific, persistent validation information. The validation information links the FQDN to both the Certificate Authority performing the validation and the specific ACME account requesting the validation.

When an ACME client accepts a "dns-persist-01" challenge, it proves control by provisioning a DNS TXT record at the Validation Domain Name. Unlike the existing "dns-01" challenge, this record is designed to persist and may be reused for multiple certificate issuances over an extended period.

3.1. Challenge Object

The challenge object for "dns-persist-01" contains the following fields:

* *type* (required, string): The string "dns-persist-01"

* *url* (required, string): The URL to which a response can be posted

* *status* (required, string): The status of this challenge

* *accounturi* (required, string): A URI identifying the ACME account requesting validation, using the identifier format specified in [RFC8657], Section 3. This is the URI the CA expects in the DNS record and the mechanism by which the CA communicates

alternative URIs to the client. Clients that pre-provisioned a record using their ACME account URL ([RFC8555], Section 7.3) SHOULD verify the value identifies the same account.

* *issuer-domain-names* (required, array of strings): A list of one or more Issuer Domain Names. The client MUST choose one of these domain names to include in the DNS TXT record. The challenge is successful if a valid TXT record is found that uses any one of the provided domain names.

  Each string in the array MUST be a domain name that complies with the following normalization rules:

  1. The domain name MUST be represented in A-label format (Punycode, [RFC5890]).

  2. All characters MUST be lowercase.

  3. The domain name MUST NOT have a trailing dot.

  The server MUST ensure the array is not empty. Servers MUST NOT send more than 10 issuer domain names. This limit serves as a practical measure to prevent denial-of-service vectors against clients. Clients MUST consider a challenge malformed if the issuer-domain-names array is empty or if it contains more than 10 entries, and MUST reject such challenges. Each domain name MUST NOT exceed 253 octets in length.

The following shows an example challenge object:

```
{
  "type": "dns-persist-01",
  "url": "https://ca.example/acme/authz/1234/0",
  "status": "pending",
  "accounturi": "https://ca.example/acct/123",
  "issuer-domain-names": ["authority.example", "ca.example.net"]
}
```

Figure 1: Example dns-persist-01 Challenge Object

4. Challenge Response and Verification

To respond to the challenge, the ACME client provisions a DNS TXT record at the Validation Domain Name of the domain being validated. The Validation Domain Name is formed by prepending the label "_validation-persist" to the domain name being validated.

For example, if the domain being validated is "example.com", the Validation Domain Name would be "_validation-persist.example.com".

The client indicates it is ready for validation by POSTing an empty JSON object ({}) to the challenge URL, following the procedure defined in [RFC8555], Section 7.5.1.

4.1. Validation Record Format

The RDATA of this TXT record MUST fulfill the following requirements:

1. The RDATA value MUST conform to the issue-value syntax defined in [RFC8659], Section 4.2. To ensure forward compatibility, the server MUST ignore any parameter within the issue-value that has an unrecognized tag.

2. The issuer-domain-name portion of the issue-value MUST be one of the Issuer Domain Names provided by the CA in the issuer-domain-names array of the challenge object. If the issuer-domain-name does not match any of the provided values, the CA MUST reject the record.

3. The issue-value MUST contain an accounturi parameter whose value is a URI that identifies the ACME account requesting validation.

   CAs MUST accept the URI of the ACME account object ([RFC8555], Section 7.3) as a valid accounturi value. CAs MAY also accept other URIs, provided each such URI uniquely and permanently identifies a single ACME account and satisfies the uniqueness requirements of [RFC8657], Section 5.4. CAs MUST NOT reassign a URI to a different account. When an ACME account is deactivated, CAs MUST treat all URIs associated with that account as invalid for validation purposes. The CA MUST verify that the URI in the DNS record identifies the ACME account making the request; if it does not, the CA MUST reject the record. In the absence of more specific comparison rules, implementations MUST use Simple String Comparison as defined in [RFC3986], Section 6.2.1.

4. The issue-value MAY contain a policy parameter. If present, this parameter modifies the validation scope. The policy parameter follows the ’tag=value’ syntax from [RFC8659]. Comparison of the values defined for this parameter MUST be case-insensitive (e.g., "WILDCARD" and "wildcard" are equivalent).

   Note: The requirement to ignore unrecognized parameters (item 1) ensures forward compatibility, allowing future extensions without breaking existing implementations, consistent with ACME’s extensibility model ([RFC8555]). The explicit case-insensitivity

requirement is necessary to ensure consistent behavior across implementations; without it, some CAs might reject unknown parameter values, preventing protocol evolution.

The following value for the policy parameter is defined with respect to subdomain and wildcard validation:

* policy=wildcard: If this value is present, the CA MAY consider this validation sufficient for issuing certificates for the validated FQDN, for specific subdomains of the validated FQDN (as covered by wildcard scope or specific subdomain validation rules), and for wildcard certificates (e.g., *.example.com). See Section 5 and Section 6.

If the policy parameter is absent, or if its value is anything other than wildcard, the CA MUST proceed as if the policy parameter were not present (i.e., the validation applies only to the specific FQDN).

5. The issue-value MAY contain a persistUntil parameter. If present, the value MUST be a base-10 encoded integer representing a UNIX timestamp (the number of seconds since 1970-01-01T00:00:00Z ignoring leap seconds). If the value is not a valid base-10 integer, the CA MUST treat the record as malformed and reject it. CAs MUST NOT consider this validation record valid for new validation attempts after the specified timestamp. However, this does not affect the reuse of already-validated data.

This specification defines the following case-handling rules for parameter values in dns-persist-01 records:

* accounturi: The value is a URI. CAs MUST compare accounturi values using Simple String Comparison per [RFC3986], Section 6.2.1, with no case-folding or other normalization.

* policy: Case-insensitive, as specified in item 4 above.

* persistUntil: The value is a base-10 integer. Case does not apply.

For example, for validation of the FQDN "example.com" with issuer domain name "authority.example" and account URI "https://ca.example/acct/123", the DNS TXT record would contain:

```
_validation-persist.example.com. IN TXT ("authority.example;"
" accounturi=https://ca.example/acct/123")
```

Figure 2: Basic Validation TXT Record

4.2. Verification Procedure

The ACME server verifies the challenge by performing a DNS lookup for TXT records at the Validation Domain Name. It then iterates through the returned records to find one that conforms to the required structure. For a record to be considered valid, its issuer-domain-name value MUST match one of the values provided in the issuer-domain-names array from the challenge object, and it MUST contain an accounturi parameter that identifies the requesting account. When comparing issuer domain names, the server MUST adhere to the normalization rules specified in Section 3.1. The server also interprets any policy parameter values according to this specification. If no record meeting all requirements is found, the server MUST treat the challenge as failed.

4.3. Multiple Issuer Support

A domain MAY authorize multiple Certificate Authorities (CAs) by provisioning a separate _validation-persist TXT record for each issuer. This allows domain owners to maintain relationships with multiple CAs simultaneously, enhancing flexibility and resilience.

4.3.1. Coexistence of Records

When multiple TXT records are present at the same DNS label (e.g., _validation-persist.example.com), each record functions as an independent authorization for the specified issuer. This follows a similar pattern to CAA records [RFC8659], where multiple records at the same label are permissible.

4.3.2. CA Verification Process

When a CA performs validation for a domain with multiple _validation-persist TXT records, it MUST follow these steps:

1. *Query DNS*: Retrieve all TXT records from the Validation Domain Name.

2. *Filter Records*: Iterate through the returned records to find those where the issuer-domain-name value matches one of the Issuer Domain Names the CA is configured to use for this validation. The CA MUST ignore all other records.

3. *Validate Record*: For each matching record, the CA proceeds to validate it according to the requirements in this specification, including verifying the accounturi and persistUntil parameters. If any matching record satisfies all requirements, the validation succeeds.

4. *Handle No Match*: If no record with a matching issuer-domain-name is found, or if no matching record satisfies all validation requirements, the validation attempt MUST fail.

4.3.3. Security and Management Considerations

When authorizing multiple issuers, domain owners MUST consider the following:

Auditing Regularly audit DNS records to ensure that only intended CAs remain authorized. Remove records for CAs that are no longer in use.

Independent Security Each authorized CA operates independently. The compromise of one CA’s systems does not directly affect the security of other authorized CAs.

Weakest Link The domain’s overall security posture is influenced by the security practices of all authorized CAs. Domain owners should consider the practices of each CA they authorize.

Authorization Removal To de-authorize a CA, the corresponding TXT record MUST be deleted from the DNS zone.

4.3.4. Example: Authorizing Two CAs

This example demonstrates how a domain owner can authorize two different CAs, "ca1.example" and "ca2.example", to issue certificates for example.org.

DNS Configuration:

```
_validation-persist.example.org. 3600 IN TXT ("ca1.example;"
" accounturi=https://ca1.example/acct/12345;"
" policy=wildcard")
_validation-persist.example.org. 3600 IN TXT ("ca2.example;"
" accounturi=https://ca2.example/acct/67890;"
" persistUntil=1767225600")
```

Figure 3: Multiple CA Authorization Records

Verification Flow for CA1:

1. CA1 queries for TXT records at _validation-persist.example.org.

2. It receives both records.

3. It filters for the record where issuer-domain-name is "ca1.example".

4. It validates the request using this record, noting the policy=wildcard authorization.

5. The second record for "ca2.example" is ignored.

Verification Flow for CA2:

1. CA2 queries for TXT records at _validation-persist.example.org.

2. It receives both records.

3. It filters for the record where issuer-domain-name is "ca2.example".

4. It validates the request using this record, noting the persistUntil constraint.

5. The first record for "ca1.example" is ignored.

4.4. Just-in-Time Validation

When processing a new authorization request, a CA MAY perform an immediate DNS lookup for _validation-persist TXT records at the Validation Domain Name corresponding to the requested domain identifier.

If one or more such records exist, the CA MUST evaluate them according to the requirements specified in Section 4.3. If at least one record meets all validation requirements, the CA MAY transition the authorization to the "valid" status without returning a "pending" challenge to the client. This mechanism is an optimization and does not alter the ACME state machine defined in [RFC8555]. The server internally transitions the authorization from "pending" through "processing" to "valid" instantaneously. From the client's perspective, it receives a "valid" authorization object directly in response to its creation request.

If no DNS TXT record meets the validation requirements, or if the records are absent, the CA MUST proceed with the standard authorization flow by returning a "pending" authorization with an associated dns-persist-01 challenge object.

CAs implementing Just-in-Time validation SHOULD rate-limit JIT DNS lookups per domain identifier, independent of the requesting account, to prevent amplification attacks where multiple accounts trigger excessive queries against a target domain. CAs SHOULD restrict JIT validation to accounts that have previously completed a successful dns-persist-01 validation. CAs implementing Just-in-Time validation SHOULD provide account activity notifications or logging, as this path eliminates the challenge-response interaction that might otherwise provide a detection window for account compromise.

This mechanism enables efficient reuse of persistent validation records while maintaining the security properties of the validation method.

4.5. Pre-Provisioning Records

Domain owners MAY provision _validation-persist TXT records before any ACME interaction occurs. When constructing records without a challenge object, the following values MUST be used:

* *accounturi*: The ACME account URL, as returned in the Location header of the account creation response ([RFC8555], Section 7.3).

* *issuer-domain-name*: Clients MAY use the caaIdentities array from the ACME directory metadata ([RFC8555], Section 7.1.1) as a hint for issuer-domain-name selection. CAs that populate caaIdentities SHOULD ensure these values are consistent with the issuer-domain-name values they accept in dns-persist-01 records. If caaIdentities is unavailable, the issuer-domain-name MAY be obtained from the CA's Certificate Policy or Certification Practice Statement.

Organizations pre-provisioning records SHOULD maintain an inventory of _validation-persist records and the ACME accounts they reference. Records MAY include a persistUntil parameter to bound their effective lifetime (see Section 7.9). Domain owners SHOULD audit _validation-persist records after any DNS infrastructure security incident, as pre-provisioned records persist beyond the window of compromise.

CAs implementing dns-persist-01 SHOULD maintain stable account URIs for the lifetime of the account and SHOULD document their URI stability guarantees. If a CA must change its URI structure, it SHOULD provide a transition period during which both old and new URIs are accepted for validation.

5. Wildcard and Subdomain Certificate Validation

This validation method supports validation for wildcard certificates (e.g., *.example.com) and specific subdomains through the use of the policy=wildcard parameter.

5.1. Scope of policy=wildcard

When a DNS TXT record includes the policy=wildcard parameter value, it authorizes certificate issuance for:

1. *The validated FQDN itself* - The base domain for which the TXT record exists (e.g., example.com)

2. *Wildcard certificates* - Wildcard certificates for the validated FQDN or any of its subdomains (e.g., *.example.com, *.dept.example.com)

3. *Specific subdomains* - Any specific subdomain of the validated FQDN (e.g., www.example.com, app.example.com, server.dept.example.com)

For example, a TXT record at _validation-persist.example.com containing policy=wildcard can validate certificates for example.com, *.example.com, www.example.com, and any other subdomain of example.com.

If the policy parameter is absent, or if its value is anything other than wildcard, the validation applies only to the specific FQDN being validated. CAs MUST NOT consider such validation sufficient for wildcard certificates or subdomains.

6. Subdomain Certificate Validation

When the policy=wildcard parameter is present (as described in Section 5), CAs MAY issue certificates for subdomains of the validated FQDN. This section describes the implementation details for subdomain validation.

6.1. Determining Permitted Subdomains

To determine which subdomains are permitted, the FQDN for which the persistent TXT record exists (referred to as the "validated FQDN") MUST be a proper suffix of the FQDN for which a certificate is requested (referred to as the "requested FQDN"). For wildcard certificate requests, the proper suffix check applies to the base domain name after removing the wildcard prefix (*.), consistent with

[RFC8555], Section 7.1.3. The base-level wildcard (e.g., *.example.com where the validated FQDN is example.com) is authorized directly by Section 5 and is not subject to this proper suffix requirement.

For example, if dept.example.com is the validated FQDN, a certificate for server.dept.example.com is permitted because dept.example.com is its suffix. A certificate for *.server.dept.example.com is also permitted: after removing the wildcard prefix, server.dept.example.com has dept.example.com as a proper suffix.

6.2. Implementation Requirements

* The persistent DNS TXT record MUST include policy=wildcard for subdomain validation to be permitted.

* CAs MUST verify that the validated FQDN is a proper suffix of the requested FQDN. For wildcard requests, this check applies to the base domain after removing the *. prefix. The base-level wildcard is exempt per Section 5.

* If the policy parameter is absent or has any value other than wildcard, subdomain validation MUST NOT be permitted.

See Section 7.3 for important security implications of enabling subdomain validation.

6.3. Example: Subdomain Validation

For a persistent TXT record provisioned at _validation-persist.example.com with policy=wildcard: - Permitted: example.com, www.example.com, app.example.com, server.dept.example.com, *.example.com, *.dept.example.com - Not permitted without additional validation: otherexample.com, example.net

7. Security Considerations

The requirement for CAs to ignore unknown parameter tags means that future extensions must be carefully designed to ensure that being ignored does not create security vulnerabilities. Extensions that require strict enforcement should use alternative mechanisms, such as separate record types or explicit version negotiation.

7.1. Persistent Record Risks

The persistence of validation records creates extended windows of vulnerability compared to traditional ACME challenge methods. If an attacker gains control of a DNS zone containing persistent validation records, they can potentially obtain certificates for the validated domains until the validation records are removed or modified.

Clients SHOULD protect validation records through appropriate DNS security measures, including:

* Using DNS providers with strong authentication and access controls
* Implementing DNS Security Extensions (DNSSEC) where possible
* Monitoring DNS zones for unauthorized changes
* Regularly reviewing and rotating validation records

7.2. Account Binding Security

The accounturi parameter provides strong binding between domain validation and specific ACME accounts. However, this binding depends on the security of the ACME account itself.

The security of this method is fundamentally bound to the security of the ACME account's private key. If this key is compromised, an attacker can immediately use any pre-existing dns-persist-01 authorizations associated with that account to issue certificates, without needing any further access to the domain's DNS infrastructure. This elevates the importance of secure key management for ACME clients far above that required for transient challenge methods, as the window of opportunity for an attacker is tied to the lifetime of the persistent authorization, not a momentary challenge.

CAs SHOULD implement robust account security measures, including:

* Strong authentication requirements for ACME accounts
* Account activity monitoring and anomaly detection
* Rapid account revocation capabilities
* Regular account security reviews
* Account key rotation policies and procedures

Clients SHOULD protect their ACME account keys with the same level of security as they would protect private keys for high-value certificates.

7.2.1. Account Key Rotation

The accounturi parameter is a stable identifier for the ACME account that persists across key rotations. When a client rotates their account key following the procedures defined in [RFC8555], Section 7.3.5, the accounturi remains unchanged. Therefore, existing DNS TXT records containing the accounturi parameter do not require modification when performing account key rotations.

7.2.2. Account URI Privacy

Because _validation-persist TXT records are publicly queryable and long-lived, the accounturi value is visible to any party that queries DNS. When the same URI appears in records for multiple domains, third parties can infer that those domains share the same ACME account and likely share infrastructure. This correlation risk is noted in [RFC8657], Section 5.9.

CAs that accept alternative URIs (see item 3 of Section 4) can mitigate this risk by issuing distinct URIs for each domain or group of domains. Such URIs SHOULD be opaque and not easily enumerable. CAs MUST protect the integrity of any URI-to-account mapping with the same controls applied to other validation infrastructure. CAs that accept alternative URIs SHOULD document their URI issuance and lifecycle policies in their Certificate Policy or Certification Practice Statement.

Domain owners who require privacy without CA cooperation can use separate ACME accounts for domains that should not be correlated. Domain owners and auditors who require independent verifiability SHOULD use the ACME account URL directly, since third parties cannot independently determine which account is bound to an alternative URI.

7.3. Subdomain Validation Risks

Enabling subdomain validation via policy=wildcard creates significant security implications. Organizations using this feature SHOULD carefully control subdomain delegation and monitor for unauthorized subdomains. This policy value serves as the explicit mechanism for domain owners to opt-in to broader validation scopes.

The ability to issue certificates for subdomains of validated FQDNs creates significant security risks, particularly in environments with subdomain delegation or where subdomains may be controlled by different entities.

Potential risks include:

* Subdomain takeover attacks where abandoned subdomains are claimed by attackers

* Unauthorized certificate issuance for subdomains controlled by different organizations

* Confusion about which entity has authority over specific subdomains

Organizations considering the use of subdomain validation SHOULD:

* Maintain strict control over subdomain delegation

* Implement monitoring for subdomain creation and changes

* Consider limiting subdomain validation to specific, controlled scenarios

* Provide clear governance policies for subdomain certificate authority

7.4. Cross-CA Validation Reuse

The persistent nature of validation records raises concerns about potential reuse across different Certificate Authorities. While the issuer-domain-name parameter is designed to prevent such reuse, implementations MUST validate that the issuer-domain-name in the DNS record matches the CA's disclosed Issuer Domain Name.

7.5. Record Tampering and Integrity

DNS records are generally not authenticated end-to-end, making them potentially vulnerable to tampering. CAs SHOULD implement additional integrity checks where possible and consider the overall security posture of the DNS infrastructure when relying on persistent validation records.

Additionally, CAs SHOULD protect their issuer-domain-name with appropriate security measures. Using DNSSEC to protect the CA's issuer-domain-name is RECOMMENDED. An attacker who compromises the DNS for a CA's issuer-domain-name could disrupt validation or

potentially impersonate the CA in certain scenarios. While this is a systemic DNS security risk that extends beyond this specification, it is amplified by any mechanism that relies on DNS for identity.

7.6. Issuer Domain Name Normalization and Limits

The issuer-domain-names field requires domain names to be provided in a normalized form (lowercase A-labels, no trailing dot) to prevent errors and security issues arising from case-sensitivity differences or Unicode homograph attacks. By requiring a canonical representation, servers and clients can perform simple byte-for-byte comparisons, ensuring interoperability and deterministic validation. The order of names in the array has no significance.

The server-side limit on the number of issuer domain names provided in a single challenge (e.g., 10) helps mitigate denial-of-service vectors where a client might be forced to perform an excessive number of DNS queries or a server might be burdened by validating against a large set of domains.

7.7. DNS Security Measures

To enhance the security and integrity of the validation process, CAs and clients should consider implementing advanced DNS security measures.

7.7.1. DNSSEC

DNS Security Extensions (DNSSEC) [RFC4033] provide cryptographic authentication of DNS data, ensuring that the validation records retrieved by a CA are authentic and have not been tampered with. CAs SHOULD use a DNSSEC-validating resolver when querying dns-persist-01 TXT records. Without one, a CA will silently accept forged responses in DNSSEC-signed zones. If a CA performs DNSSEC validation, it MUST treat validation failure (e.g., expired signatures, broken chain of trust) as a challenge failure and MUST NOT use the record for domain validation. This requirement is stricter than the general DNSSEC guidance in [RFC8555] because dns-persist-01 records are long-lived and their compromise would persist for the record’s lifetime.

7.7.2. Multi-Perspective Validation

Multi-Perspective Issuance Corroboration (MPIC) is a technique to validate domain control from multiple network vantage points. This is a critical defense against localized network attacks, such as BGP hijacking and DNS spoofing, which could otherwise lead to certificate mis-issuance.

For CAs subject to requirements like the CA/Browser Forum Baseline Requirements, MPIC is essential for robust domain validation. However, for private PKI systems where the network topology is well-known and such localized attacks are not part of the threat model, operators might reasonably judge MPIC unnecessary.

7.8. Validation Data Reuse and TTL Handling

This validation method is explicitly designed for persistence and reuse. The period for which a CA may rely on validation data is its Validation Data Reuse Period (as defined in Section 2). However, if the DNS TXT record's Time-to-Live (TTL) is shorter than this period, the CA MUST treat the record's TTL as the effective validation data reuse period for that specific validation. A TTL of zero means the CA MUST NOT reuse the validation data beyond the current validation attempt.

CAs MAY reuse validation data obtained through this method for the duration of their validation data reuse period, subject to the TTL constraints described in this section. CAs MUST also respect any persistUntil constraint as specified in Section 4.1.

7.9. persistUntil Parameter Considerations

The persistUntil parameter provides domain owners with direct control over the validity period of their validation records. CAs and clients should be aware of the following considerations:

* Domain owners should set expiration dates for validation records that balance security and operational needs. To avoid unexpected validation failures during certificate renewal, domain owners are advised to:
  - Align persistUntil values with certificate lifetimes or planned maintenance intervals
  - Monitor or set reminders for persistUntil expirations
  - Document persistUntil practices in certificate management procedures
  - Automate updates to validation records with new persistUntil values during certificate renewal workflows
* CAs MUST parse and apply the persistUntil timestamp as specified in Section 4.1.

7.10. Revocation and Invalidation of Persistent Authorizations

The persistent nature of dns-persist-01 authorizations means that a valid DNS TXT record can grant control for an extended period, potentially even if the domain owner's intent changes or if the associated ACME account key is compromised. Therefore, explicit mechanisms for revoking or invalidating these persistent authorizations are critical.

The primary method for an Applicant to invalidate a dns-persist-01 authorization for a domain is to *remove the corresponding DNS TXT record* from the Validation Domain Name. After the record is removed and resolver caches have expired, new validation attempts for the domain will fail. This behavior represents a deliberate design trade-off: any existing authorization obtained via this method will remain valid until it expires as per the CA's Validation Data Reuse Period. This persistence underscores the importance of protecting the ACME account key.

For situations requiring immediate revocation of issuance capability, such as a suspected account key compromise, the primary and most effective mechanism is to *deactivate the ACME account* as specified in [RFC8555], Section 7.3.6. Deactivating the account immediately and irrevocably prevents it from being used for any further certificate issuance.

ACME Clients SHOULD provide clear mechanisms for users to:

* Remove the _validation-persist DNS TXT record.

* Monitor the presence and content of their _validation-persist records to ensure they accurately reflect desired authorization.

Certificate Authorities (CAs) implementing this method MUST:

* During a validation attempt, fail the validation if the corresponding DNS TXT record is no longer present or if its content does not meet the requirements of this specification (e.g., incorrect issuer-domain-name, missing accounturi, altered policy).

* Respect the persistUntil constraint as specified in Section 4.1, rejecting new validation attempts after the specified timestamp even if the record remains present.

* Ensure their internal systems are capable of efficiently handling the validation failure when DNS records are removed or become invalid.

While this method provides a persistent signal of control, the fundamental ACME authorization object (as defined in [RFC8555]) remains subject to its own lifecycle, including expiration. A persistent DNS record allows for repeated authorizations, but each authorization object issued by the CA will have a defined validity period, after which it expires unless renewed.

8. IANA Considerations

8.1. ACME Validation Methods Registry

IANA is requested to register the following entry in the "ACME Validation Methods" registry:

* *Label*: dns-persist-01
* *Identifier Type*: dns
* *ACME*: Y
* *Reference*: This document

8.2. Underscored and Globally Scoped DNS Node Names Registry

IANA is requested to register the following entry in the "Underscored and Globally Scoped DNS Node Names" registry defined in [RFC8552]:

* *RR Type*: TXT
* *_NODE NAME*: _validation-persist
* *Reference*: This document

9. Implementation Considerations

When designing future extensions to this specification, new parameters SHOULD be designed to degrade gracefully when ignored by CAs that do not recognize them. Parameters that fundamentally change the security properties of the validation SHOULD NOT be introduced without a version negotiation mechanism.

9.1. DNS Record Size Considerations

The RDATA of the TXT record, which contains the issue-value, may become large, particularly if the accounturi is long. While the total size of a TXT record's RDATA can be up to 65,535 octets, it must be formatted as a sequence of one or more character-strings, where each string is limited to 255 octets in length.

CA Implementation Guidelines: - CAs SHOULD endeavor to keep the accounturi values they generate reasonably concise to minimize the final record size.

Client Implementation Guidelines: - Clients MUST properly handle the creation of TXT records where the RDATA exceeds 255 octets. As specified in [RFC1035], Section 3.3.14, clients MUST split the RDATA into multiple, concatenated, quote-enclosed strings, each no more than 255 octets. For example:

```
~~~ dns
_validation-persist.example.com. IN TXT ("first-part-of-long-string..."
" ...second-part-of-long-string")
~~~
{: #ex-long-txt-record title="Multi-String TXT Record Format"}
```

Failure to correctly format long RDATA values may result in validation failures.

9.2. Domain Name Normalization Algorithm

This section provides an algorithm for domain name normalization to promote interoperability. Both clients and servers SHOULD follow a consistent normalization process to ensure that domain names are handled uniformly.

The recommended normalization process consists of the following four steps, applied in order:

1. *Case-folding*: Apply Unicode-aware, locale-independent case-folding to the entire domain name string to convert it to lowercase.

2. *Unicode Normalization*: Normalize the string to Unicode Normalization Form C (NFC).

3. *Punycode Conversion*: Convert each label of the domain name to its A-label (Punycode) representation as specified in [RFC5890].

4. *Trailing Dot Removal*: Remove any trailing dot from the final string.

For example, a domain name like EXAMPLE.com. is normalized as follows: 1. After case-folding: example.com. 2. After NFC normalization: example.com. 3. After Punycode conversion: example.com. 4. After removing trailing dot: example.com

An internationalized domain name like テシテ選CODE-example.com. is normalized as follows: 1. After case-folding: テシnicode-example.com. 2. After NFC normalization: テシnicode-example.com. 3. After Punycode conversion: xn--nicode-example-9jb.com. 4. After removing trailing dot: xn--nicode-example-9jb.com

9.3. CA Implementation Guidelines

Certificate Authorities implementing this validation method should consider:

* Establishing clear policies for Issuer Domain Name disclosure in Certificate Policies and Certification Practice Statements

* Developing procedures for handling validation record TTL variations

* Creating account security monitoring and incident response procedures

* Providing clear documentation for clients on proper record construction

9.3.1. Error Handling

When implementing the "dns-persist-01" validation method, Certificate Authorities SHOULD return appropriate ACME error codes to provide clear feedback on validation failures. Specifically:

* CAs SHOULD return a malformed error (as defined in [RFC8555]) when the TXT record has invalid syntax, such as duplicate parameters, invalid timestamp format in the persistUntil parameter, missing mandatory accounturi parameter, or other syntactic violations of the record format specified in this document.

* CAs SHOULD return an unauthorized error (as defined in [RFC8555]) when validation fails due to authorization issues, including:

  - The accounturi parameter in the DNS TXT record does not identify the ACME account making the request

- The persistUntil timestamp has expired, indicating that the validation record is no longer considered valid for new validation attempts

- The issuer-domain-name in the DNS TXT record does not match any of the values provided in the issuer-domain-names array of the challenge object

Note that these error codes apply to validation attempts on specific challenges. In the case of Just-in-Time Validation (see Section 4.4), when a CA finds a pre-existing DNS TXT record that does not meet validation requirements, the CA proceeds with the standard authorization flow by issuing a new pending challenge rather than returning an error.

These error codes help ACME clients distinguish between different types of validation failures and take appropriate corrective actions.

9.4. Client Implementation Guidelines

ACME clients implementing this validation method should consider:

* Implementing secure DNS record management practices

* Providing clear user interfaces for managing persistent validation records

* Implementing validation record monitoring and alerting

* Designing appropriate error handling for validation failures

* Considering the security implications of persistent records in their threat models

9.5. DNS Provider Considerations

DNS providers supporting this validation method should consider:

* Implementing appropriate access controls for validation record management

* Providing audit logging for validation record changes

* Supporting reasonable TTL values for validation records

* Considering dedicated interfaces or APIs for ACME validation record management

10. Examples

10.1. Basic Validation Example (FQDN Only)

For validation of "example.com" by a CA using "authority.example" as its Issuer Domain Name, where the validation should only apply to "example.com":

1. CA provides challenge object with a list of valid Issuer Domain Names:

```
{
  "type": "dns-persist-01",
  "url": "https://ca.example/acme/authz/1234/0",
  "status": "pending",
  "accounturi": "https://ca.example/acct/123",
  "issuer-domain-names": ["authority.example", "ca.example.net"]
}
```

2. Client chooses one of the provided Issuer Domain Names (e.g., "authority.example") and provisions a DNS TXT record (note the absence of a policy parameter for scope):

```
_validation-persist.example.com. IN TXT ("authority.example;"
" accounturi=https://ca.example/acct/123")
```

3. CA validates the record through DNS queries. This validation is sufficient only for "example.com".

10.2. Wildcard Validation Example

For validation of "*.example.com" (which also validates "example.com" and specific subdomains like "www.example.com") by a CA using "authority.example" as its Issuer Domain Name:

1. The CA provides a challenge object similar to the basic example, containing an issuer-domain-names array.

2. Client chooses one of the provided Issuer Domain Names (e.g., "authority.example") and provisions a DNS TXT record at the base domain’s Validation Domain Name, including policy=wildcard:

```
_validation-persist.example.com. IN TXT ("authority.example;"
" accounturi=https://ca.example/acct/123;"
" policy=wildcard")
```

Figure 4: Wildcard Policy Validation Record

3. CA validates the record through DNS queries. This validation authorizes certificates for "example.com", "*.example.com", and specific subdomains like "www.example.com".

10.3. Validation Example with persistUntil

For validation of "example.com" with an explicit expiration date:

1. The CA provides a challenge object similar to the basic example, containing an issuer-domain-names array.

2. Client chooses one of the provided Issuer Domain Names (e.g., "authority.example") and provisions a DNS TXT record including persistUntil:

```
_validation-persist.example.com. IN TXT ("authority.example;"
" accounturi=https://ca.example/acct/123;"
" persistUntil=1721952000")
```

Figure 5: Validation Record with Expiration Time

3. CA validates the record. This validation is sufficient only for "example.com" and will not be considered valid after the specified timestamp (2024-07-26T00:00:00Z).

10.4. Wildcard Validation Example with persistUntil

For validation of "*.example.com" with an explicit expiration date:

1. The CA provides a challenge object similar to the basic example, containing an issuer-domain-names array.

2. Client chooses one of the provided Issuer Domain Names (e.g., "authority.example") and provisions a DNS TXT record including policy=wildcard and persistUntil:

```
_validation-persist.example.com. IN TXT ("authority.example;"
" accounturi=https://ca.example/acct/123;"
" policy=wildcard;"
" persistUntil=1721952000")
```

Figure 6: Wildcard Validation Record with Expiration Time

3. CA validates the record. This validation authorizes certificates for "example.com", "*.example.com", and specific subdomains, but will not be considered valid after the specified timestamp (2024-07-26T00:00:00Z).

11. References

11.1. Normative References

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC8552] Crocker, D., "Scoped Interpretation of DNS Resource Records through "Underscored" Naming of Attribute Leaves", BCP 222, RFC 8552, DOI 10.17487/RFC8552, March 2019, <https://www.rfc-editor.org/info/rfc8552>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

11.2. Informative References

[birgelee-sc082-security]
Birge-Lee, H., "Security of SC-082 Redux", 2025.

[cabf-br] "Baseline Requirements for the Issuance and Management of Publicly-Trusted TLS Server Certificates", 2025, <https://cabforum.org/baseline-requirements-documents/>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

Acknowledgments

The authors acknowledge prior community work that directly informed this specification:

* The CA/Browser Forum ballot proposals to enable persistent / static DNS Domain Control Validation signals in the Baseline Requirements [cabf-br], in particular Ballot SC-082 ("Clarify CA Assisted DNS Validation under 3.2.2.4.7", authored by Michael Slaughter) and the active proposal SC-088 ("DNS TXT Record with Persistent Value DCV Method", also authored by Michael Slaughter). These efforts provided the policy framing and initial industry discussion motivating standardization of a reusable ACME DNS validation record.

* The formal and empirical security analysis of static / persistent DCV methods performed by Henry Birge-Lee ("Proof of static DCV security" presentation, the "Security of SC-082 Redux" paper [birgelee-sc082-security], and related research), which helped clarify the threat model and informed the security considerations in this document.

* The Delegated DNS Domain Validation (DDDV) Threat Modeling Tiger Team discussions and document ("Validation SC - Delegated DNS Domain Validation (DDDV) Threat Model"), whose participants contributed to broad threat enumeration; notable contributors include Michael Slaughter (Amazon Trust Services), Corey Bonnell (DigiCert), Clint Wilson (Apple), and Martijn Katerbarg (Sectigo).

The authors also thank members of the ACME Working Group and CA/Browser Forum who provided early review, critique, and operational perspectives on persistent validation records.

Any errors or omissions are the responsibility of the authors.

Authors' Addresses

Shiloh Heurich
Fastly
Email: sheurich@fastly.com

Henry Birge-Lee
Crosslayer Labs, Inc.
Email: henry@crosslayerlabs.com

Michael Slaughter
Amazon Trust Services
Email: slghtr@amazon.com