Stream: Internet Engineering Task Force (IETF)
RFC: 9448
Category: Standards Track
Published: September 2023
ISSN: 2070-1721
Authors: C. Wendt, *Somos Inc.*; D. Hancock, *Somos Inc.*; M. Barnes, *Neustar Inc.*; J. Peterson, *Neustar Inc.*

RFC 9448
TNAuthList Profile of Automated Certificate Management Environment (ACME) Authority Token

Abstract

This document defines a profile of the Automated Certificate Management Environment (ACME) Authority Token for the automated and authorized creation of certificates for Voice over IP (VoIP) telephone providers to support Secure Telephone Identity (STI) using the TNAuthList defined by STI certificates.

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 https://www.rfc-editor.org/info/rfc9448.

Table of Contents

1. Introduction

[RFC8555] describes a mechanism for automating certificate management on the Internet. It enables administrative entities to prove effective control over resources like domain names, and it automates the process of generating and issuing certificates. [RFC9447] extends ACME to provide a general method of extending the authority and authorization of entities to control a resource via a third party Token Authority beyond the certification authority (CA).

This document is a profile document using the Authority Token mechanism defined in [RFC9447]. It is a profile that specifically addresses the Secure Telephone Identity Revisited (STIR) problem statement described in [RFC7340], which identifies the need for Internet credentials that can attest authority for the originator of VoIP calls in order to detect impersonation, which is currently an enabler for common attacks associated with illegal robocalling, voicemail hacking, and swatting. These credentials are used to sign Personal Assertion Tokens (PASSporTs) [RFC8225], which can be carried in using protocols such as SIP [RFC8224]. Currently, the only defined credentials for this purpose are the certificates specified in [RFC8226] using the TNAuthList. This document defines the use of the TNAuthList Authority Token in the ACME challenge to prove the authoritative use of the contents of the TNAuthList, including a Service Provider Code (SPC), a telephone number, or a set of telephone numbers or telephone number blocks.

This document also describes the ability for a telephone authority to authorize the creation of CA types of certificates for delegation, as defined in [RFC9060].

2. Requirements Language

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.

3. ACME New-Order Identifiers for TNAuthList

Section 7 of [RFC8555] defines the procedure that an ACME client uses to order a new certificate from a CA. The new-order request contains an identifier field that specifies the identifier objects the order corresponds to. This document defines a new type of identifier object called TNAuthList. A TNAuthList identifier contains the identity information to be populated in the TNAuthList of the new certificate. For the TNAuthList identifier, the new-order request includes a type set to the string "TNAuthList". The value of the TNAuthList identifier **MUST** be set to the details of the TNAuthList requested.

The string that represents the TNAuthList **MUST** be constructed using base64url encoding, as described in Section 5 of [RFC4648] and as defined in Section 2 of JSON Web Signature [RFC7515]. The base64url encoding **MUST NOT** include any padding characters, and the TNAuthList ASN.1 object **MUST** be encoded using DER encoding rules.

An example of an ACME order object "identifiers" field containing a TNAuthList certificate is as follows:

```
 "identifiers": [{"type":"TNAuthList","value":"F83n2a...avn27DN3"}]
```

where the "value" object string represents the arbitrary length of the base64url-encoded string.

A full new-order request would look as follows:

```
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":"TNAuthList","value":"F83n...n27DN3"}],
    "notBefore": "2021-01-01T00:00:00Z",
    "notAfter": "2021-01-08T00:00:00Z"
  }),
  "signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g"
}
```

On receiving a valid new-order request, the ACME server creates an authorization object ([RFC8555], Section 7.1.4), containing the challenge that the ACME client must satisfy to demonstrate authority for the identifiers specified by the new order (in this case, the TNAuthList identifier). The CA adds the authorization object URL to the "authorizations" field of the order object and returns the order object to the ACME client in the body of a 201 (Created) response.

```
HTTP/1.1 201 Created
Content-Type: application/json
Replay-Nonce: MYAuvOpaoIiywTezizk5vw
Location: https://example.com/acme/order/1234

{
  "status": "pending",
  "expires": "2022-01-08T00:00:00Z",

  "notBefore": "2022-01-01T00:00:00Z",
  "notAfter": "2022-01-08T00:00:00Z",
  "identifiers":[{"type":"TNAuthList",
                  "value":"F83n2a...avn27DN3"}],

  "authorizations": [
   "https://example.com/acme/authz/1234"
  ],
  "finalize": "https://example.com/acme/order/1234/finalize"
}
```

4. TNAuthList Identifier Authorization

On receiving the new-order response, the ACME client queries the referenced authorization object to obtain the challenges for the identifier contained in the new-order request, as shown in the following example request and response.

```
POST /acme/authz/1234 HTTP/1.1
    Host: example.com
    Content-Type: application/jose+json

    {
      "protected": base64url({
        "alg": "ES256",
        "kid": " https://example.com/acme/acct/evOfKhNU60wg",
        "nonce": "uQpSjlRb4vQVCjVYAyyUWg",
        "url": "https://example.com/acme/authz/1234"
      }),
      "payload": "",
      "signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"
    }
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Link: <https://example.com/acme/some-directory>;rel="index"

{
  "status": "pending",
  "expires": "2022-01-08T00:00:00Z",

  "identifier": {
    "type":"TNAuthList",
    "value":"F83n2a...avn27DN3"
  },

  "challenges": [
    {
      "type": "tkauth-01",
      "tkauth-type": "atc",
      "token-authority": "https://authority.example.org",
      "url": "https://example.com/acme/chall/prV_B7yEyA4",
      "token": "IlirfxKKXAsHtmzK29Pj8A"
    }
  ]
}
```

When processing a certificate order containing an identifier of type "TNAuthList", a CA uses the Authority Token challenge type of "tkauth-01" with a "tkauth-type" of "atc" in [RFC9447] to verify that the requesting ACME client has authenticated and authorized control over the requested resources represented by the "TNAuthList" value.

The challenge "token-authority" parameter is only used in cases where the VoIP telephone network requires the CA to identify the Token Authority. This is currently not the case for the Signature-based Handling of Asserted information using toKENs (SHAKEN) [ATIS-1000080] certificate framework governance but may be used by other frameworks. If a "token-authority" parameter is present, then the ACME client **MAY** use the "token-authority" value to identify the URL representing the Token Authority that will provide the TNAuthList Authority Token response to the challenge. If the "token-authority" parameter is not present, then the ACME client **MUST** identify the Token Authority based on locally configured information or local policies.

The ACME client responds to the challenge by posting the TNAuthList Authority Token to the challenge URL identified in the returned ACME authorization object, an example of which follows:

```
POST /acme/chall/prV_B7yEyA4 HTTP/1.1
Host: boulder.example.com
Content-Type: application/jose+json

{
  "protected": base64url({
  "alg": "ES256",
  "kid": "https://example.com/acme/acct/evOfKhNU60wg",
  "nonce": "Q_s3MWoqT05TrdkM2MTDcw",
  "url": "https://boulder.example.com/acme/authz/asdf/0"
  }),
  "payload": base64url({
  "tkauth": "DGyRejmCefe7v4N...vb29HhjjLPSggwiE"
  }),
  "signature": "9cbg5JO1Gf5YLjjz...SpkUfcdPai9uVYYQ"
}
```

The "tkauth" field is defined as a new field in the challenge object specific to the tkauth-01 challenge type that should contain the TNAuthList Authority Token defined in the next section.

5. TNAuthList Authority Token

The TNAuthList Authority Token is a profile instance of the ACME Authority Token defined in [RFC9447].

The TNAuthList Authority Token protected header **MUST** comply with "Request Authentication" (Section 6.2 of [RFC8555]).

The TNAuthList Authority Token Payload **MUST** include the mandatory claims "exp", "jti", and "atc" and **MAY** include the optional claims defined for the Authority Token detailed in the next subsections.

5.1. "iss" Claim

The "iss" claim is an optional claim defined in [RFC7519], Section 4.1.1. It can be used as a URL identifying the Token Authority that issued the TNAuthList Authority Token beyond the "x5u" or other header claims that identify the location of the certificate or certificate chain of the Token Authority used to validate the TNAuthList Authority Token.

5.2. "exp" Claim

The "exp" claim, defined in [RFC7519], Section 4.1.4, **MUST** be included and contains the DateTime value of the ending date and time that the TNAuthList Authority Token expires.

5.3. "jti" Claim

The "jti" claim, defined in [RFC7519], Section 4.1.7, **MUST** be included and contains a unique identifier for this TNAuthList Authority Token transaction.

5.4. "atc" Claim

The "atc" claim **MUST** be included and is defined in [RFC9447]. It contains a JSON object with the following elements:

- a "tktype" key with a string value equal to "TNAuthList" to represent a TNAuthList profile of the Authority Token [RFC9447] defined by this document. "tktype" is a required key and **MUST** be included.
- a "tkvalue" key with a string value equal to the base64url encoding of the TNAuthList certificate extension ASN.1 object using DER encoding rules. "tkvalue" is a required key and **MUST** be included.
- a "ca" key with a boolean value set to either true when the requested certificate is allowed to be a CA cert for delegation uses or false when the requested certificate is not intended to be a CA cert, only an end-entity certificate. "ca" is an optional key; if not included, the "ca" value is considered false by default.
- a "fingerprint" key constructed as defined in [RFC8555], Section 8.1, corresponding to the computation of the "Thumbprint" step using the ACME account key credentials. "fingerprint" is a required key and **MUST** be included.

An example of the TNAuthList Authority Token is as follows:

```
{
  "protected": base64url({
    "typ":"JWT",
    "alg":"ES256",
    "x5u":"https://authority.example.org/cert"
  }),
  "payload": base64url({
    "iss":"https://authority.example.org",
    "exp":1640995200,
    "jti":"id6098364921",
    "atc":{"tktype":"TNAuthList",
      "tkvalue":"F83n2a...avn27DN3",
      "ca":false,
      "fingerprint":"SHA256 56:3E:CF:AE:83:CA:4D:15:B0:29:FF:1B:71:
       D3:BA:B9:19:81:F8:50:9B:DF:4A:D4:39:72:E2:B1:F0:B9:38:E3"}
  }),
  "signature": "9cbg5JO1Gf5YLjjz...SpkUfcdPai9uVYYQ"
}
```

5.5. Acquiring the Token from the Token Authority

Following [RFC9447], Section 5, the Authority Token should be acquired using a RESTful HTTP POST transaction as follows:

```
POST /at/account/:id/token HTTP/1.1
Host: authority.example.org
Content-Type: application/json
```

The request will pass the account identifier as a string in the request parameter "id". This string will be managed as an identifier specific to the Token Authority's relationship with a Communications Service Provider (CSP). There is assumed to also be a corresponding authentication procedure that can be verified for the success of this transaction, for example, an HTTP authorization header containing valid authorization credentials, as defined in [RFC9110], Section 11.6.2.

The body of the POST request **MUST** contain a JSON object with key value pairs corresponding to values that are requested as the content of the claims in the issued token. As an example, the body **SHOULD** contain a JSON object as follows:

```
{
  "tktype":"TNAuthList",
  "tkvalue":"F83n2a...avn27DN3",
  "ca":false,
  "fingerprint":"SHA256 56:3E:CF:AE:83:CA:4D:15:B0:29:FF:1B:71:D3
    :BA:B9:19:81:F8:50:9B:DF:4A:D4:39:72:E2:B1:F0:B9:38:E3"
}
```

If successful, the response to the POST request returns a 200 (OK) with a JSON body that contains, at a minimum, the TNAuthList Authority Token as a JSON object with a key of "token" and the base64url-encoded string representing the atc token. JSON is easily extensible, so users of this specification may want to pass other pieces of information relevant to a specific application.

An example of a successful response would be as follows:

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

{"token": "DGyRejmCefe7v4N...vb29HhjjLPSggwiE"}
```

If the request is not successful, the response should indicate the error condition. Specifically, for the case that the authorization credentials are invalid or if the account identifier provided does not exist, the response code **MUST** be 403 (Forbidden). Other 4xx and 5xx responses **MUST** follow standard HTTP error condition conventions [RFC9110].

5.6. Token Authority Responsibilities

When creating the TNAuthList Authority Token, the Token Authority **MUST** validate that the information contained in the ASN.1 TNAuthList accurately represents the service provider code (SPC) or telephone number (TN) resources the requesting party is authorized to represent based on their pre-established, verified, and secure relationship between the Token Authority and the

requesting party. Note that the fingerprint in the token request is not meant to be verified by the Token Authority but rather is meant to be signed as part of the token so that the party that requests the token can, as part of the challenge response, allow the ACME server to validate that the token requested and used came from the same party that controls the ACME client.

5.7. Scope of the TNAuthList

Because this specification specifically involves the TNAuthList defined in [RFC8226], which involves SPC, telephone number ranges, and individual telephone numbers, the client may also request an Authority Token with some subset of its own authority as the TNAuthList provided in the "tkvalue" element in the "atc" JSON object. Generally, the scope of authority representing a CSP is represented by a particular SPC (e.g., in North America, an operating company number (OCN) or service provider identifier (SPID)). Based on number allocations, that provider is also generally associated with a particular set of different telephone number ranges and/or telephone numbers. The TNAuthList can be constructed to define a limited scope of the TelephoneNumberRanges or TelephoneNumbers ([RFC8226], Section 9) either associated with an SPC or with the scope of telephone number ranges or telephone numbers the client has authority over.

As recommended in the Security Considerations section in [RFC9447], an Authority Token can either have a scope that attests all of the resources that a client is eligible to receive certificates for or potentially a more limited scope that is intended to capture only those resources for which a client will receive a certificate from a particular certification authority. Any certification authority that sees an Authority Token can learn information about the resources a client can claim. In cases where this incurs a privacy risk, Authority Token scopes should be limited to only the resources that will be attested by the requested ACME certificate.

6. Validating the TNAuthList Authority Token

Upon receiving a response to the challenge, the ACME server **MUST** perform the following steps to determine the validity of the response.

1. Verify that the value of the "atc" claim is a well-formed JSON object containing the mandatory key values.
2. If there is an "x5u" parameter, verify the "x5u" parameter is an HTTPS URL with a reference to a certificate representing the trusted issuer of Authority Tokens for the ecosystem.
3. If there is an "x5c" parameter, verify the certificate array contains a certificate representing the trusted issuer of Authority Tokens for the ecosystem.
4. Verify the TNAuthList Authority Token signature using the public key of the certificate referenced by the token's "x5u" or "x5c" parameter.
5. Verify that "atc" claim contains a "tktype" identifier with the value "TNAuthList".
6. Verify that the "atc" claim "tkvalue" identifier contains the equivalent base64url-encoded TNAuthList certificate extension string value as the identifier specified in the original challenge.
7. Verify that the remaining claims are valid (e.g., verify that token has not expired).

8. Verify that the "atc" claim "fingerprint" is valid and matches the account key of the client making the request.
9. Verify that the "atc" claim "ca" identifier boolean corresponds to the CA boolean in the Basic Constraints extension in the Certificate Signing Request (CSR) for either CA certificate or end-entity certificate.

If all steps in the token validation process pass, then the ACME server **MUST** set the challenge object "status" to "valid". If any step of the validation process fails, the "status" in the challenge object **MUST** be set to "invalid".

7. Using ACME-Issued Certificates with JSON Web Signature

JSON Web Signature (JWS) [RFC7515] objects can include an "x5u" header parameter to refer to a certificate that is used to validate the JWS signature. For example, the STIR PASSporT framework [RFC8225] uses "x5u" to indicate the STIR certificate used to validate the PASSporT JWS object. The URLs used in "x5u" are expected to provide the required certificate in response to a GET request, not a POST-as-GET, as required for the "certificate" URL in the ACME order object. Thus, the current mechanism generally requires the ACME client to download the certificate and host it on a public URL to make it accessible to relying parties. This section defines an optional mechanism for the certification authority (CA) to host the certificate directly and provide a URL that the ACME client owner can directly reference in the "x5u" of their signed PASSporTs.

As described in Section 7.4 of [RFC8555], when the certificate is ready for making a "finalize" request, the server will return a 200 (OK) with the updated order object. In this response, an ACME server can add a newly defined field called "x5u" that can pass this URL to the ACME client for usage in generated PASSporTs as a publicly available URL for PASSporT validation.

x5u (optional, string): a URL that can be used to reference the certificate in the "x5u" parameter of a JWS object [RFC7515]

The publishing of the certificates at the new "x5u" URL should follow the GET request requirement as mentioned above and should be consistent with the timely publication according to the durations of the certificate life cycle.

The following is an example of the use of "x5u" in the response when the certificate status is "valid".

```
HTTP/1.1 200 OK
Content-Type: application/json
Replay-Nonce: CGf81JWBsq8QyIgPCi9Q9X
Link: <https://example.com/acme/directory>;rel="index"
Location: https://example.com/acme/order/TOlocE8rfgo

{
  "status": "valid",
  "expires": "2016-01-20T14:09:07.99Z",

  "notBefore": "2016-01-01T00:00:00Z",
  "notAfter": "2016-01-08T00:00:00Z",

  "identifiers": [
    "type":"TNAuthList",
    "value":"F83n2a...avn27DN3"
  ],

  "authorizations": ["https://sti-ca.com/acme/authz/1234"],

  "finalize": "https://example.com/acme/order/TOlocE8rfgo/finalize",

  "certificate": "https://example.com/acme/cert/mAt3xBGaobw",

  "x5u": "https://example.com/cert-repo/giJI53km23.pem"
}
```

8. Usage Considerations

8.1. Large Number of Noncontiguous TNAuthList Values

There are many scenarios and reasons to have various combinations of SPCs, TNs, and TN ranges. [RFC8226] has provided a somewhat unbounded set of combinations. It's possible that a complex noncontiguous set of telephone numbers are being managed by a CSP. Best practice may be simply to split a set of noncontiguous numbers under management into multiple STI certificates to represent the various contiguous parts of the greater noncontiguous set of TNs, particularly if the length of the set of values in an identifier object grows to be too large.

9. IANA Considerations

Per this document, IANA has added a new identifier object type to the "ACME Identifier Types" registry defined in Section 9.7.7 of [RFC8555].

Label	Reference
TNAuthList	RFC 9448

Table 1

10. Security Considerations

The token represented by this document has the credentials to represent the scope of a telephone number, a block of telephone numbers, or an entire set of telephone numbers represented by an SPC. The creation, transport, and any storage of this token **MUST** follow the strictest of security best practices beyond the recommendations of the use of encrypted transport protocols in this document to protect it from getting in the hands of bad actors with illegitimate intent to impersonate telephone numbers.

This document inherits the security properties of [RFC9447]. Implementations should follow the best practices identified in [RFC8725].

This document only specifies SHA256 for the fingerprint hash. However, the syntax of the fingerprint object would permit other algorithms if, due to concerns about algorithmic agility, a more robust algorithm were required at a future time. Future specifications can define new algorithms for the fingerprint object as needed.

11. References

11.1. 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>.

[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>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[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>.

[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>.

[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>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/info/rfc8725>.

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

[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>.

[RFC9447] Peterson, J., Barnes, M., Hancock, D., and C. Wendt, "Automated Certificate Management Environment (ACME) Challenges Using an Authority Token", RFC 9447, DOI 10.17487/RFC9447, August 2023, <https://www.rfc-editor.org/info/rfc9447>.

11.2. Informative References

[ATIS-1000080] ATIS, "Signature-based Handling of Asserted information using toKENs (SHAKEN): Governance Model and Certificate Management", ATIS-1000080.v005, December 2022, <https://access.atis.org/apps/group_public/download.php/69428/ATIS-1000080.v005.pdf>.

[RFC7340] Peterson, J., Schulzrinne, H., and H. Tschofenig, "Secure Telephone Identity Problem Statement and Requirements", RFC 7340, DOI 10.17487/RFC7340, September 2014, <https://www.rfc-editor.org/info/rfc7340>.

[RFC8224] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 8224, DOI 10.17487/RFC8224, February 2018, <https://www.rfc-editor.org/info/rfc8224>.

[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>.

Acknowledgements

We would like to thank Richard Barnes and Russ Housley for valuable contributions to this document.

Authors' Addresses

Chris Wendt
Somos Inc.
United States of America
Email: chris-ietf@chriswendt.net

David Hancock
Somos Inc.
United States of America
Email: davidhancock.ietf@gmail.com

Mary Barnes
Neustar Inc.
United States of America
Email: mary.ietf.barnes@gmail.com

Jon Peterson
Neustar Inc.
Suite 570
1800 Sutter St
Concord, CA 94520
United States of America
Email: jon.peterson@neustar.biz