
| | |
|------------------|---|
| Workgroup: | Internet Engineering Task Force |
| Internet-Draft: | draft-weidner-catalog-rr-ext-00 |
| Published: | 3 June 2025 |
| Intended Status: | Informational |
| Expires: | 5 December 2025 |
| Author: | M. S. Weidner, Ed. <i>Centurion Intelligence Consulting Agency</i> |

CATALOG Authorization Transparency And Log Overlay for Graph-based DNS

Abstract

This document proposes an extension to the Certification Authority Authorization (CAA) DNS Resource Record (RR) that enables the mandatory or optional binding of Certificate Transparency (CT) Log URIs directly within DNS. By embedding CT-Log endpoints in CAA RR, Certification Authorities (CAs) gain a standardized, discoverable mechanism for retrieving preferred and permitted CT-Log endpoint information, thereby enhancing the security and auditability of X.509 TLS certificate issuance. The extension defines the syntax and semantics for a new CAA property tag, **"issuect"**, and introduces a parameter set consisting of *"desc"*, *"critical"*, *"validfrom"*, *"validtill"*, *"cturi"*, *"logid"*, and *"pubkey"*. It outlines validation and processing rules, discusses deployment considerations, privacy implications, and interoperability with existing DNS, CA, and CT infrastructures. Additionally, the draft proposes an MTA-STS-like hardening mechanism, called **CAA-CT-STS**, for the new property to further strengthen the PKI ecosystem. Finally, it introduces the recursive acronym CATALOG (CATALOG Authorization Transparency And Log Overlay for Graph-based DNS) as a mnemonic for the overall extension framework.

Note

TO BE REMOVED: This document is being collaborated on in Gitea at: <https://git.coresecret.dev/msw/draft-weidner-catalog-rr-ext.git> [1] The most recent working version of this document, open issues, and related resources are available there. The author gratefully accepts pull requests. The author's PGP key is available at: [7A83 41E5 F157 0319 D80F 4418 A11E 8851 9A3D 8CF6](#) [2].

The list of current Internet-Drafts can be accessed in plain text at:
<https://www.ietf.org/1id-abstracts.html>

The list of current Internet-Drafts can also be accessed at:
<https://datatracker.ietf.org/drafts/current/>

The list of Internet-Draft Shadow Directories can be accessed at:
<https://www.ietf.org/shadow.html>

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 5 December 2025.

Copyright Notice

Copyright (c) 2025 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 | 5 |
| 1.1. Certification Authority Authorization | 5 |
| 1.2. Certificate Transparency | 5 |
| 1.3. CATALOG approach | 5 |
| 1.3.1. CAA "issuance" Property Tag | 5 |
| 1.3.2. CAA Certificate Transparency Strict Transport Security (CAA-CT-STS) | 6 |
| 1.3.3. Operational discussions | 6 |
| 2. Terminology | 7 |
| 2.1. Keywords | 7 |
| 2.2. Abbreviations and Definitions | 7 |
| 2.3. CAA Syntax conformity | 9 |

| | |
|--|----|
| 3. CAA "issuect" Property Tag | 9 |
| 3.1. CAA "issuect" Property Tag Definitions | 9 |
| 3.1.1. CAA "issuect" Property Tag and its Parameters Syntax | 9 |
| 3.1.2. CAA "issuect" Property Tag Canonical Presentation Format | 12 |
| 3.1.3. CAA "issuect" Property Tag and its Parameters Definition, Meaning and Order | 12 |
| 3.1.4. Special Case: Empty "issuect" Parameter | 14 |
| 3.2. CAA "issuect" Property Tag Processing | 14 |
| 3.2.1. Pre-issuance Requirements | 14 |
| 3.2.2. Prior to Issuing a Certificate | 15 |
| 3.2.3. Runtime Processing | 15 |
| 4. CAA Certificate Transparency Strict Transport Security (CAA-CT-STS) | 16 |
| 4.1. CAA-CT-STS Definitions | 16 |
| 4.1.1. DNS "_caa-ct-sts" TXT RR | 16 |
| 4.1.2. DNS "_caa-ct-sts" TXT RR Syntax | 16 |
| 4.1.3. CAA-CT-STS Policy File | 17 |
| 4.1.4. CAA-CT-STS Policy File Syntax | 17 |
| 4.2. CAA-CT-STS Processing | 19 |
| 4.2.1. CAA-CT-STS Policy File HTTPS Fetching | 19 |
| 4.2.2. CAA-CT-STS Policy File Validation | 19 |
| 5. IANA Considerations | 20 |
| 5.1. IANA registration CAA Property Tag "issuect" | 20 |
| 5.1.1. IANA Subregistry Entry "issuect" | 20 |
| 5.2. Well-Known URIs Registry | 20 |
| 5.2.1. IANA Registry Entry URI Suffix "caa-ct-sts.txt" | 20 |
| 5.3. CAA-CT-STS Strict Transport Security Registry Creation | 21 |
| 5.3.1. CAA-CT-STS Record Fields | 21 |
| 5.3.2. CAA-CT-STS Record Policy Fields | 21 |
| 6. Security Considerations | 21 |
| 6.1. Global considerations | 22 |
| 6.1.1. CAs Preferred Proposed Validation Path | 22 |

| | |
|--|----|
| 6.2. CAA-CT-STS Advantages and Disadvantages | 23 |
| 6.3. Policy Redundancy Considerations $ C \geq n + 1 \wedge W \leq 2$ | 23 |
| 6.3.1. Operational Procedure for CAs | 24 |
| 6.4. Scope Limited to CAs and CT-Log Operators Processing CAA RRs | 24 |
| 6.5. Additional CAA-CT-STS Security Considerations | 24 |
| 6.6. Authorization Freshness | 25 |
| 6.7. Use with and without DNSSEC | 26 |
| 6.8. Scenario Effects for "issuect" and Security | 26 |
| 6.9. Restrictions Supersedable by DNS Delegation | 26 |
| 7. References | 27 |
| 7.1. Normative References | 27 |
| 7.2. Informative References | 29 |
| 7.3. URI | 30 |
| Appendix A. Informational: CAA "issuect" Property Tag Processing Checklist | 30 |
| Appendix B. Informational: DNS Zone-file Examples | 32 |
| B.1. Preface | 32 |
| B.2. CAA "issuect" Comprehensive and Proper Example | 32 |
| B.3. CAA "issuect" No CT-Log Submission Example | 34 |
| B.4. CAA "issuect" Erroneous Example | 34 |
| B.5. CAA-CT-STS "_caa-ct-sts" Example | 34 |
| Appendix C. Informational: CAA-CT-STS Examples | 35 |
| C.1. CAA-CT-STS Policy File Example | 35 |
| Appendix D. Informational: POSIX compliant Scripts | 36 |
| D.1. Shell "issuect" Generator script | 36 |
| D.2. Shell "caa-ct-sts.txt" Generator script | 39 |
| Appendix E. Informational: DNS TTL | 41 |
| E.1. High-Security, High-Change Environments | 41 |
| E.2. Stable, Low-Risk Domains with DDoS Concerns | 41 |
| E.3. Balanced Approach for Most | 41 |

| | |
|---|----|
| Appendix F. Informational: Change History | 41 |
| F.1. draft-weidner-catalog-rr-ext-00 | 41 |
| Acknowledgements | 41 |
| Contributors | 41 |
| Author's Address | 42 |

1. Introduction

1.1. Certification Authority Authorization

Certification Authority Authorization (CAA) [RFC8659] RRs empower domain owners to specify, via DNS [RFC1035], which Certificate Authorities (CAs) in a Public Key Infrastructure [RFC5280] may or may not issue Certificates for their domains.

Similarly, the CAA RR Extensions for Account URI and Automatic Certificate Management Environment (ACME) Method Binding [RFC8657] extend the CAA grammar to introduce additional parameters for CAA RRs. These extensions enable or disable specified validation methods and bind a domain to specific accounts authorized to submit Certificate Signing Requests (CSRs) and retrieve end-entity Certificates.

Additionally, the Certification Authority Authorization (CAA) processing for Email Addresses [RFC9495] extends the established CAA RR mechanism by introducing a new property, "issuemail", to apply CAA controls to the issuance of S/MIME [RFC8551] Certificates.

1.2. Certificate Transparency

Furthermore, Certificate Transparency (CT) Version 1 [RFC6962] (still in use), and Certificate Transparency (CT) Version 2 [RFC9162] (which obsoletes RFC6962), provide public, append-only ledgers of issued certificates, enabling domain owners and relying parties to detect misissuance.

1.3. CATALOG approach

Currently, there is no standardized, discoverable mechanism in DNS for a domain owner to declare, which Certificate Transparency (CT) Logs must or may record its Certificates. As a result, CAs rely on out-of-band configurations or hard-coded lists, increasing operational complexity and expanding the attack surface.

1.3.1. CAA "issuect" Property Tag

This extension introduces a new CAA Property Tag, "issuect", Section 3, with the following parameters: "critical", "desc", "validfrom", "validtill", "cturi", "logid", and "pubkey". Domain owners can embed as many CT-Log endpoints directly in their DNS zones. Each CAA "issuect" RR names a single URI for log submission and retrieval, specifies the time window during which whitelisted

CAs may or must submit entries, and enumerates the authorized CT-Logs Public Keys. Because "**issuect**" lives alongside existing CAA tags in a DNSSEC [RFC9364] protected zone CAs can discover, validate, and enforce CT-Log policies without additional out-of-band trust anchors. To enforce secure-by-default behavior, "**issuect**" supports explicit whitelisting combined with default-deny semantics. Domain owners list only approved CT-Logs; any CT-Log not so listed is implicitly excluded. Furthermore, a special blacklist flag can be set, preventing all CT-Log operations unless explicitly permitted. An optional critical flag (when true) mandates that no Certificate may be issued without being logged to the specified CT-Log endpoint. These dual mechanisms guard against accidental use of unvetted logs, and mitigate risks from CT-Log Operator compromise, preventing downgrade or surprise attacks.

Google, for instance, maintains a publicly available "List of Trustworthy CT-Log Operators", CT-Truststore, including its [Known CT-Logs \[3\]](#), and a [JSON list of Logs compliant with Chrome's CT Policy \[4\]](#). Apple similarly publishes its own CT-Truststore, detailed in its [Certificate Transparency Policy \[5\]](#), and a [JSON list of Logs meeting Apple's CT requirements \[6\]](#).

By republishing and allowlisting selected entries from these audited CT-Log lists, and any future community-maintained CT-Truststore, within the same DNS zone that hosts a domain's CAA RRs, domain owners gain a single, authoritative repository for:

- (a) CA authorization,
- (b) CA authorization metadata (e.g., validation methods and ACME account bindings),
- (c) CA authorization for S/MIME Certificate issuance,
- (d) CT-Log authorization.

Embedding curated CT-Log trust anchors directly in DNS consolidates trust management, reduces external dependencies for CAs, minimizes lookup latency, preserves privacy by limiting out-of-band queries, and surfaces misconfigurations or tampering at DNSSEC validation time.

1.3.2. CAA Certificate Transparency Strict Transport Security (CAA-CT-STTS)

CAA-CT-STTS, [Section 4](#), is a policy framework that combines the mechanisms of CAA, CT, and MTA-STTS by which a domain holder can publish Certificate-Transparency enforcement requirements alongside CA Authorization (CAA) over a second secured channel. It mirrors the approach of SMTP MTA-STTS [RFC8461], using DNS TXT RR, a specific subdomain and an HTTPS only served policy file. A special DNS TXT record "**_caa-ct-sts**" [Section 4.1.1](#) lets clients discover the existence and version of the policy, and a policy file, retrieved from "*https://caa-ct-sts.<domain>/.well-known/caa-ct-sts.txt*", contains structured rules in a canonical key:value format. The before mentioned section defines the discovery mechanism, DNS TXT RR and policy format (with ABNF), and procedures for fetching, validating, and applying CAA-CT-STTS policies.

1.3.3. Operational discussions

Lastly, practical aspects such as a CA Processing Checklist, [Appendix A](#), caching behavior [Appendix E](#), and fallback, [Section 6.1.1](#), strategies for non-DNSSEC zones are addressed. Throughout this document, the framework is referred to by the recursive acronym **CATALOG** (CATALOG Authorization Transparency And Log Overlay for Graph-based DNS).

The following sections specify the complete syntax and semantics of both of the CAA **"issuect"** Property Tag, [Section 3](#), and CAA-CT-STS framework, [Section 4](#), detail processing rules for CAs and resolvers for both of the CAA **"issuect"** Property Tag, [Section 3.2](#), and CAA-CT-STS, [Section 4.2](#), framework.

2. Terminology

2.1. Keywords

The keywords **"MUST"**, **"MUST NOT"**, **"REQUIRED"**, **"SHALL"**, **"SHALL NOT"**, **"SHOULD"**, **"SHOULD NOT"**, **"RECOMMENDED"**, **"NOT RECOMMENDED"**, **"MAY"**, and **"OPTIONAL"** in this document are to be interpreted as described in BCP 14 [\[RFC2119\]](#), [\[RFC8174\]](#) when, and only when, they appear in all capitals, as shown here.

2.2. Abbreviations and Definitions

| | | |
|-------------------|--|---|
| CA | Certification Authority | An Issuer that issues Certificates in accordance with a specified Certificate Policy (CP). |
| CAA | Certification Authority Authorization | Allows a DNS domain name holder to specify one or more CAs authorized to issue Certificates for that domain name. See: [RFC8657] , [RFC8659] . |
| CAA-CT-STS | CAA Certificate Transparency Strict Transport Security | CAA-CT-STS is a policy framework, by which a domain holder can publish Certificate-Transparency enforcement requirements alongside CA authorization (CAA). See: Section 4 . |
| CAA-CT-STS Policy | CAA-CT-STS Policy File | The set of rules fetched via HTTPS from the Policy Domains <code>"/.well-known/caa-ct-sts.txt"</code> . See: Section 4.1.3 . |
| CATALOG | CATALOG Authorization Transparency And Log Overlay for Graph-based DNS | A Proposal to enrich DNS-based CAA RRs with Certificate Transparency URI bindings. This document: [URI1] . |
| CA-TS | Certification Authority Truststore | A CA-TS is the set of root certificates from Certificate Authorities that a platform (e.g., Mozilla, Apple, Microsoft, or the Java runtime) pre-loads and implicitly trusts when TLS Certificates are validated. It defines, which CAs are accepted by default and is maintained through regular audits and policy checks to remove or distrust any CA that fails to meet security or operational requirements. |

| | | |
|--------|--|--|
| CP | Certificate Policy | Specifies the criteria that a CA undertakes to meet in its issue of Certificates. See: [RFC3647] . |
| CPS | Certification Practices Statement | Specifies the means by which the criteria of the CP are met. In most cases, this will be the document against which the operations of the CA are audited. See: [RFC3647] . |
| CT | Certification Transparency | Certificate Transparency aims to mitigate the problem of misissued Certificates by providing append-only logs of issued Certificates. See: [RFC6962] , [RFC9162] . |
| CT-TS | Certificate Transparency Truststore | A Certificate Transparency Truststore is a curated whitelist of public Certificate Transparency Logs, maintained primarily by Apple and Google, that are accepted as valid append-only repositories for issued TLS certificates. By requiring certificates to be submitted to one of these approved logs, the CT-TS ensures how could be verified that a Certificate has been publicly recorded, making misissuance or hidden Certificates easier to detect. |
| DANE | DNS-Based Authentication of Named Entities | TLSA RRs associate a Certificate or a public key of an end-entity or a trusted issuing authority with the corresponding Transport Layer Security (TLS) [RFC5246] or Datagram Transport Layer Security (DTLS) [RFC6347] transport endpoint. See: [RFC6698] , [RFC7671] . |
| DNS | Domain Name System | The Internet naming system specified in [RFC1034] , [RFC1035] , [RFC9499] , and any revisions to these specifications. |
| DNSSEC | DNS Security Extensions | Extensions to the DNS that provide authentication services as specified in [RFC4033] , [RFC4034] , [RFC4035] , [RFC5155] , [RFC9364] , and any revisions to these specifications. |

| | | |
|---------|------------------------------------|---|
| DT | DNSSEC DS RR Transparency | DNSSEC Transparency applies the Certificate Transparency model to DNSSEC by logging every Delegation Signer (DS) RR in an append-only Merkle tree, enabling cryptographic proofs of inclusion and consistent auditing of DNSSEC key delegations. This ensures that any unauthorized or missing DS updates become publicly detectable, strengthening trust in the DNS delegation chain. (This is still in the early stages of development before a first I-D is released.) |
| MTA-STS | SMTP MTA Strict Transport Security | SMTP MTA Strict Transport Security (MTA-STS) is a mechanism enabling mail service providers (SPs) to declare their ability to receive Transport Layer Security (TLS) secure SMTP connections [RFC8461]. |
| RR | DNS Resource Record | A particular entry in the DNS, including the owner name, class, type, time to live, and data, as defined in [RFC1034], [RFC2181]. |

Table 1: Abbreviations and Definitions

2.3. CAA Syntax conformity

All syntax introduced in this draft is strictly in accordance with the CAA RR framework defined in [RFC8659], and [RFC8657] when and only when **NOT** stated otherwise.

3. CAA "issuect" Property Tag

3.1. CAA "issuect" Property Tag Definitions

3.1.1. CAA "issuect" Property Tag and its Parameters Syntax

This document defines the CAA "issuect" Property Tag.

The presence of one or more CAA "issuect" Properties in the relevant CAA Resource Record Set (RRSet) indicates that the domain is requesting that Certification Authorities restrict the submission of Certificates to specified Certification Transparency Logs only.

The CAA "issuect" Property Tag has the following sub-syntax (specified in ABNF as per [RFC5234]).

The production rules for

- "ALPHA",
- "DIGIT",
- "DQUOTE",
- "SP",

are defined in [Appendix B.1](#) of [\[RFC5234\]](#).

In addition, **absolute-URI** is defined in [Section 4.3](#) of [\[RFC3986\]](#).

```

ALPHA      = %x 4 1 - 5 A / %x 6 1 - 7 A      ; A-Z / a-z
DIGIT      = %x 3 0 - 3 9                      ; 0 - 9
SP         = %x 2 0                          ; SPACE
DQUOTE     = %x 2 2                          ; "
HEXDIG     = DIGIT / %x 4 1 - 4 6 / %x 6 1 - 6 6 ; 0 - 9 / A-F / a-f

issuect-property = "issuect" SP DQUOTE issuect-value DQUOTE

issuect-value   = issuer-domain-name ";" SP
                  critical-param ";" SP
                  desc-param ";" SP
                  validfrom-param ";" SP
                  validtill-param ";" SP
                  cturi-param ";" SP
                  logid-param ";" SP
                  pubkey-param ";"

issuer-domain-name = label * ( "." label )
label              = ( ALPHA / DIGIT ) * ( " - " ( ALPHA / DIGIT ) )

critical-param    = "critical" "=" ( "true" / "false" )

desc-param        = "desc" "=" "" desc-text ""
desc-text         = * ( %x 2 0 - 2 6 / %x 2 8 - 7 E ) ; printable ASCII without: '

validfrom-param   = "validfrom" "=" date-time
validtill-param   = "validtill" "=" date-time

date-time         = date "T" time "Z"
date              = 4 DIGIT "-" 2 DIGIT "-" 2 DIGIT
time              = 2 DIGIT ":" 2 DIGIT ":" 2 DIGIT

cturi-param       = "cturi" "=" uri
uri               = absolute-URI

logid-param       = "logid" "=" "" b 6 4 -string ""

pubkey-param      = "pubkey" "=" "" b 6 4 -string ""

b 6 4 -char       = ALPHA / DIGIT / "+" / "/"
b 6 4 -string     = 1 * ( b 6 4 -char / "=" )

```

Figure 1: CAA "issuect" Property Tag ABNF Syntax

The meanings of each production rule within the issuect-value grammar are as follows:

issuer-domain-name

: A Certification Authority's domain name, composed of one or more labels.

label

: A single domain label consisting only of ASCII letters, digits, or hyphens (an "LDH label").

critical-param

: The parameter name critical, followed by "=", followed by the boolean value true or false.

desc-param

: The parameter name desc, followed by "=", followed by a desc-text value enclosed in single quotes.

desc-text

: A string of printable ASCII characters (letters, digits, and symbols), excluding the single-quote character (').

validfrom

: The parameter name validfrom, followed by "=", followed by a date-time value.

validtill

: The parameter name validtill, followed by "=", followed by a date-time value.

date-time

: A date value, the letter "T", a time value, then the letter "Z".

date

: Four digits, a hyphen, two digits, another hyphen, and two digits (i.e., YYYY-MM-DD).

time

: Two digits, a colon, two digits, another colon, and two digits (i.e., hh:mm:ss).

cturi-param

: The parameter name cturi, followed by "=", followed by an uri.

uri

: An absolute URI as defined by [\[RFC3986\]](#).

logid-param

: The parameter name logid, followed by "=", followed by a b64-string enclosed in single quotes.

pubkey-param

: The parameter name pubkey, followed by "=", followed by a b64-string enclosed in single quotes.

b64-string

: One or more Base64 characters (b64-char), optionally ending with "=" padding.

b64-char

: A single character drawn from uppercase and lowercase letters (A – Z, a–z), digits (0–9), plus ("+"), slash ("/"), or hyphen ("-").

3.1.2. CAA "issuect" Property Tag Canonical Presentation Format

The canonical presentation format of the CAA "issuect" Property Tag is:

```
issuect "<issuer-domain>; <critical>; <description>; <validfrom>; <validtill>; <uri>; <logid>;  
<pubkey>;"
```

Figure 2: Canonical Presentation Format

- (a) The tag "issuect" **MUST** always appear in lowercase.
- (b) The complete property parameter set **MUST** be enclosed in double quotes.
- (c) A single space **MUST** appear after "issuect" and after each semicolon, except for the final semicolon after the last parameter, where no space follows.
- (d) All domain labels and hostnames **MUST** be in lowercase.
- (e) The value of the "critical" flag **MUST** be either "true" or "false".
- (f) The string following "desc=" **MUST** be enclosed in single quotes, and **MUST NOT** contain any single quote character (") within the string itself.
- (g) Timestamps **MUST** strictly follow the format "YYYY-MM-DDThh:mm:ssZ".
- (h) The URI **MUST** remain unchanged, except that all hostnames within it **MUST** be lowercase, in accordance with DNS naming conventions.
- (i) The string following "logid=" **MUST** be enclosed in single quotes and **MUST** contain the BASE64-encoded SHA-256 hash of the CT-Log's DER-encoded SubjectPublicKeyInfo.
- (j) The string following "pubkey=" **MUST** be enclosed in single quotes and **MUST** contain the BASE64-encoded ASN.1-format SubjectPublicKeyInfo of the CT-Log itself.
- (k) A trailing semicolon **MUST** follow the "pubkey" parameter.

Whereas

- Timestamps are according to [\[ISO-8601\]](#).
- BASE64-encoding is according to [\[RFC4648\]](#).
- SHA256-hash is according to [\[RFC6234\]](#).
- DER-encoding is according to [\[X.690\]](#).
- ASN.1-format is according to [\[X.680\]](#).

3.1.3. CAA "issuect" Property Tag and its Parameters Definition, Meaning and Order

The CAA "issuect" Property Tag contains exactly eight elements, which **MUST** appear in the order listed below. Each element **MUST** be terminated by a semicolon (;) and a single space, except where noted.

- (1) **Issuer Domain Name**

The authorized Certification Authority's domain name, in LDH notation (ASCII letters, digits, hyphens), as defined in Section 3.1.

- Position: #1
- Syntax: issuer-domain-name;

(2) **critical**

A boolean flag enforcing mandatory logging.

- Position: #2
- Value: "true" or "false"
- Syntax: critical=<true|false>;

(3) **desc**

A human-readable description of the CT-Log.

- Position: #3
- Value: any printable ASCII string excluding the single-quote character ('), enclosed in single quotes
- Syntax: desc='<desc-text>;

(4) **validfrom**

The start of the CT-Log acceptance window.

- Position: #4
- Value: UTC timestamp in strict ISO 8601 format (YYYY-MM-DDThh:mm:ssZ)
- Syntax: validfrom=<YYYY-MM-DDThh:mm:ssZ>;

(5) **validtill**

The end of the CT-Log acceptance window.

- Position: #5
- Value: UTC timestamp in strict ISO 8601 format (YYYY-MM-DDThh:mm:ssZ)
- Syntax: validtill=<YYYY-MM-DDThh:mm:ssZ>;

(6) **cturi**

The absolute URI for CT-Log submissions and queries.

- Position: #6
- Value: any URI conforming to [\[RFC3986\]](#), with hostnames in lowercase
- Syntax: cturi=<absolute-URI>;

(7) **logid**

The CT-Log's identifier.

- Position: #7
- Value: Base64-encoded SHA-256 hash of the CT-Log's DER-encoded SubjectPublicKeyInfo, enclosed in single quotes
- Syntax: logid='<BASE64-SHA256>';

(8) **key**

The CT-Log's public key.

- Position: #8
- Value: Base64-encoded ASN.1-format SubjectPublicKeyInfo of the CT-Log, enclosed in single quotes
- Syntax: key='<BASE64-PUBKEY>';

3.1.4. Special Case: Empty "issuect" Parameter

A CAA "issuect" RR containing only the terminating semicolon (;) and no parameters **MUST** be interpreted as a prohibition on all CT-Log submissions for the specified domain. In other words:

```
issuect ";
```

Figure 3: Special Case: Empty "issuect" Parameter

indicates that no CT-Logs may be used. Because CAA authorizations are additive, if multiple CAA "issuect" RRs exist, any non-empty CAA "issuect" RR (with actual parameters) **MUST** take precedence over an empty one. Thus, the effective set of permitted CT-Logs is the union of all non-empty CAA "issuect" RRs, and an empty CAA "issuect" RR contributes no additional authorizations.

3.2. CAA "issuect" Property Tag Processing

3.2.1. Pre-issuance Requirements

Prior to issuing a Certificate, a Certification Authority **MUST** obtain at least $n + 1$ unique CAA "issuect" RRs for each CA account, where n is the minimum number of Signed Certificate Timestamps (SCTs) [Section 3.2](#) as per [\[RFC6962\]](#). The value of n **MAY** vary based on the requested Certificate's validity period, CT-Log Operator Policies, the CA's CP/CPS, and the CA/Browser Forum Baseline Requirements. The additional " $n + 1$ " RR provides redundancy to tolerate one CT-Log failure or unavailability.

Note: Determining the exact minimum number of unique CAA **"issuect"** RRs per CA is outside the scope of this document. Domain owners **SHOULD** refer to the CT-Log policies published by major operators, (e.g., [Apple's Certificate Transparency Policy \[5\]](#), and [Chrome's Certificate Transparency Policy \[7\]](#), and to the CP/CPS of the issuing CA for specific requirements.

3.2.2. Prior to Issuing a Certificate

Before issuing a Certificate that certifies a domain, the Certification Authority **MUST** check for the publication of a relevant RRSset. Discovery of the relevant RRsset **MUST** be performed according to the algorithm specified in [Section 3](#) of [\[RFC8659\]](#). If the relevant RRsset is empty or does not contain any **"issuect"** Properties, it is interpreted that the domain owner has not requested any restrictions regarding the submission to specific CT-Logs. If the Certificate certifies multiple domains (for example, in the Subject Alternative Name extension), the Certification Authority **MUST** perform the above procedure separately for each domain being certified. The presence of an **"issuect"** Property **MUST NOT** replace or supersede the required validation of the domain name as specified by the **"issue"** or **"issuewild"** Properties in the CAA RRsset, as outlined in the CAA specification [\[RFC8659\]](#). Certification Authorities **MUST** still validate authorization according to the CAA **"issue"** and / or **"issuewild"** policies. For example, if a CAA **"issue"** Property authorizes "CA A" to issue Certificates, but the **"issuect"** Property references a CT-Log belonging to "CA B", this results in an unsatisfiable configuration. In such cases, issuance **MUST NOT** proceed. Furthermore, if the **"issuect"** Parameter Set is incorrectly formatted or contains invalid values, it **MUST** be treated as non-existent.

3.2.3. Runtime Processing

The internal handling of **"issuect"** parameters is left to each Certification Authority's implementation and is outside the scope of this document.

However, the following requirements **MUST** be met:

(1) ABNF Conformance

- Any CAA **"issuect"** RR whose parameters do not conform to the ABNF grammar in [Section 3](#) **MUST** be treated as though the **"issuect"** tag does not exist.

(2) Malformed RRs

- If a RR is syntactically malformed (e.g., missing parameters, incorrect ordering, invalid quoting), it **MUST** be ignored in its entirety.

(3) CT-Log Policy Enforcement

- After parsing all valid CAA **"issuect"** RRs, the CA **MUST** verify that the resulting set of whitelisted CT-Logs satisfies its own CT-Log policy, (as published in its CP/CPS, CT-Log operator requirements, or other policy documents).

- If the CA's policy cannot be met, for example, because too few CT-Logs are whitelisted, Certificate issuance **MUST** fail.

4. CAA Certificate Transparency Strict Transport Security (CAA-CT-STS)

4.1. CAA-CT-STS Definitions

4.1.1. DNS "_caa-ct-sts" TXT RR

The DNS "_caa-ct-sts" TXT RR is a DNS TXT RR named "_caa-ct-sts" at the Policy domain.

It must be encoded in US-ASCII and consists of semicolon-separated key/value pairs. The following fields are defined:

- v (plaintext, required): protocol version. The only valid value is "CAACTSTSV1".
- id (plaintext, required): a version identifier string (1–32 alphanumeric chars) used to signal updates. This must change whenever the policy file is updated.
- Extension fields (optional): any other key:value pairs (see ABNF below).

The record must begin with the "v="-field. The "id="-field is used by clients to detect policy updates (similar to MTA-STS).

If DNS returns multiple "_caa-ct-sts" TXT RR, or if none of them starts with "v=CAACTSTSV1", or if parsing fails, clients **MUST** assume no **CAA-CT-STS** policy is available. DNS CNAME RR are allowed: if "_caa-ct-sts" is a CNAME to another "_caa-ct-sts" RR, clients follow the alias (as in MTA-STS).

4.1.2. DNS "_caa-ct-sts" TXT RR Syntax

The DNS "_caa-ct-sts" TXT RR content has the following sub-syntax (specified in ABNF as per [\[RFC7405\]](#)):


```
issuect-text-record = issuect-version 1 * (issuect-field-delim issuect-field) \
    [issuect-field-delim]

issuect-field      = issuect-id / issuect-extension

issuect-field-delim = * WSP ";" * WSP

issuect-version    = %s"v=CAACTSTSV1"

issuect-id         = %s"id=" 1 * 3 2 (ALPHA / DIGIT)

issuect-extension  = issuect-ext-name "=" issuect-ext-value

issuect-ext-name   = (ALPHA / DIGIT) * 3 1 (ALPHA / DIGIT / "_" / "-" / ".")

issuect-ext-value  = 1 * (%x 2 1 - 3 A / %x 3 C / %x 3 E- 7 E)
    ; printable chars excluding "=" ";" and space
```

Figure 4: ABNF Syntax of "_caa-ct-sts" TXT RR

This mirrors the MTA-STS TXT ABNF. Clients accept a single TXT string (concatenating fragments if needed) and ignore extra semicolons. The id string has no semantics beyond change detection; it imposes no ordering.

4.1.3. CAA-CT-STS Policy File

Once a "_caa-ct-sts" RR indicates support, the "CAA-CT-STS" policy is fetched via HTTPS contains newline-separated "key:value"-pairs, similar to MTA-STS. The canonical policy fields are:

- (a) version (required): **MUST** be "CAACTSTSV1". This identifies the policy version.
- (b) max_age (required): a non-negative integer (seconds) giving the lifetime of the policy. Clients **SHOULD** cache the policy up to max_age (up to 31 557 600).
- (c) ct_policy (required): at least one. For each CT-Log a single entry has to be provided according to the same syntax as defined in [Section 3.1.1](#) while ignoring the leading "issuect" Tag, see: [Appendix C](#) for examples.
- (d) Extension fields: additional key:value pairs allowed.

All fields appear as <key>:[space]*<value> on separate lines.

4.1.4. CAA-CT-STS Policy File Syntax

The `"/.well-known/caa-ct-sts.txt"` Policy File has the following sub-syntax (specified in ABNF as per [RFC7405](#)):

```

ALPHA      = %x 4 1 - 5 A / %x 6 1 - 7 A ; A-Z / a-z
DIGIT      = %x 3 0 - 3 9 ; 0 - 9
SP         = %x 2 0 ; SPACE
DQUOTE     = %x 2 2 ; "
SEMICOLON  = %x 3 B ; ";"
LPAREN     = %x 2 8 ; "("
RPAREN     = %x 2 9 ; ")"
CRLF      = %x 0 D. 0 A / %x 0 A ; CRLF or LF

b 6 4 -char = ALPHA / DIGIT / "+" / "/"
b 6 4 -string = 1 * ( b 6 4 -char / "=" )

date       = 4 DIGIT "-" 2 DIGIT "-" 2 DIGIT
time       = 2 DIGIT ":" 2 DIGIT ":" 2 DIGIT
date-time  = date "T" time "Z"

uri        = absolute-URI

issuect-value = issuer-domain-name ";" SP
              "critical" "=" ( "true" / "false" ) ";" SP
              "desc" "=" desc-text ";" SP
              "validfrom" "=" date-time ";" SP
              "validtill" "=" date-time ";" SP
              "cturi" "=" uri ";" SP
              "logid" "=" b 6 4 -string ";" SP
              "pubkey" "=" b 6 4 -string ";"

issuer-domain-name = label * ( "." label )
label              = ( ALPHA / DIGIT ) * ( ( "-" ) ( ALPHA / DIGIT ) )
desc-text          = * ( %x 2 0 - 2 6 / %x 2 8 - 7 E )
                  ; printable ASCII except apostrophe

;-----
; CAA-CT-STS Policy File Grammar
;-----

caa-ct-sts-file = * WSP
                 version-line CRLF
                 mode-line CRLF
                 max-age-line CRLF
                 1 * ct-policy-line
                 * WSP

version-line     = "version:" SP "CAACTSTSv 1 "
max-age-line     = "max_age:" SP 1 * DIGIT

ct-policy-line  = "ct_policy:" SP
                 LPAREN SP
                 DQUOTE issuect-value DQUOTE
                 SP RPAREN

```

```
WSP      = * ( SP / HTAB )
```

Figure 5: ABNF Syntax of `"/.well-known/caa-ct-sts.txt"` Policy File

In the above, each policy field occupies an entire line and policy fields may appear in any order. Parsers **MUST** accept policies that are syntactically valid; unknown fields are treated as extensions and ignored. A policy with no "ct_log" entries (and "mode" not explicitly set to "none") is invalid. See Section TODO for policy validation.

4.2. CAA-CT-STs Processing

4.2.1. CAA-CT-STs Policy File HTTPS Fetching

The CAA-CT-STs Policy File is fetched via "HTTPS GET" via TLS1.3 [RFC8446], only, from

```
https://caa-ct-sts.<domain>.<tld>/.well-known/caa-ct-sts.txt
```

Figure 6: HTTPS GET `"/.well-known/caa-ct-sts.txt"` Policy File

where <domain> is the Policy Domain (e.g., example.com). This follows the well-known URI convention. The Policy Server **MUST** present a Certificate, that is valid for the "caa-ct-sts" DNS-ID (e.g., "caa-ct-sts.example.com"), chain to a Root CA that is trusted by the Certification Authority, which is asked to issue a new Certificate, as the Certification Authority rely on TLS security for policy integrity. Certification Authorities **SHOULD** verify that the HTTP Content-Type is "text/plain" (ignoring any parameters); other content types indicate an invalid policy file. Line separators must be either LF or CRLF; the file **MUST** be US-ASCII or UTF-8 without a byte-order mark. No authentication is performed beyond standard TLS trust. After fetching, Certification Authorities parse the file as above. If the HTTP request fails for whatever reason, (network error, invalid cert, status \neq 200, or parse error), the policy is considered unavailable or invalid, and Certification Authorities fall back to "no policy". HTTP 3xx redirects **MUST NOT** be followed, and HTTP caching [RFC7234] **MUST NOT** be used. A new or updated Policy is identified when its content differs (e.g., a new "id" in DNS or new "version" in the Policy File); while the "max_age" field controls how long a policy may be cached. As in MTA-STs, Certification Authorities **SHOULD** check for updated policy via DNS before permanently acting on failures.

4.2.2. CAA-CT-STs Policy File Validation

Before use, a retrieved Policy File by a Certification Authority **MUST** be checked for correct syntax and semantics:

- The "version" field **MUST** equal "CAACTSTSV1",
- While "max_age" **MUST** be a valid non-negative integer,
- and at least "1 + n" "ct_log" **MUST** be present, see: [Section 3.2.1](#).

If the policy file is malformed or has invalid values, it is rejected (treated as if no policy were present), as with MTA-STS.

Lastly the Runtime Processing as described in [Section 3.2.3](#) is applicable, too.

5. IANA Considerations

5.1. IANA registration CAA Property Tag "issuect"

The following IANA registration is requested.

According to [\[RFC8126\]](#) these information are provided:

- Registry: Certification Authority Restriction Properties.
- Registry Group: Public Key Infrastructure using X.509 (PKIX) Parameters.
- Expert Contact: Mr. Phillip Hallam-Baker (at time of writing).
- URI: [Public Key Infrastructure using X.509 \(PKIX\) Parameters \[8\]](#).

IANA is asked to assign the property tag "**issuect**" to this subregistry, documenting its name, ABNF syntax, and reference to this specification.

5.1.1. IANA Subregistry Entry "issuect"

| Tag | Meaning | Reference |
|---------|----------------------------------|-----------------|
| issuect | Authorized CT-Log submission URI | (This document) |

Table 2: IANA Subregistry Entry "issuect"

5.2. Well-Known URIs Registry

The following IANA registration is requested.

According to [\[RFC8126\]](#) these information are provided:

- Registry: Well-Known URIs
- Registry Group: Well-Known URIs.
- Expert Contact: Mr. Mark Nottingham (at time of writing).
- URI: [Well-Known URIs \[9\]](#).

IANA is asked to assign the "well-known" URI Suffix "**caa-ct-sts.txt**" to this registry, whose Change Controller is: IETF

5.2.1. IANA Registry Entry URI Suffix "caa-ct-sts.txt"

| URI Suffix | Reference | Change Controller |
|----------------|-----------------|-------------------|
| caa-ct-sts.txt | (This document) | IETF |

Table 3: IANA Registry Entry URI Suffix "caa-ct-sts.txt"

5.3. CAA-CT-STS Strict Transport Security Registry Creation

The following IANA "Registry" registration is requested.

According to [RFC8126] these information are provided:

- Registry: CAA-CT-STS Strict Transport Security (CAA-CT-STS) [to be created]
- Subregistry: CAA-CT-STS TXT Record Fields [to be created]
- Subregistry: CAA-CT-STS TXT Policy Fields [to be created]
- Expert Contact: (tba).

IANA is asked to create the before mentioned Registry.

5.3.1. CAA-CT-STS Record Fields

| Field name | Description | Reference |
|------------|--------------------|---------------|
| v | Record version | This document |
| id | Policy instance ID | This document |

Table 4: CAA-CT-STS Record Fields

5.3.2. CAA-CT-STS Record Policy Fields

| Field name | Description | Reference |
|------------|-------------------------------|---------------|
| version | Policy version | This document |
| mode | Enforcement behavior | This document |
| max_age | Policy lifetime | This document |
| ct_policy | Unique CT-Log property string | This document |

Table 5: CAA-CT-STS Policy Fields

6. Security Considerations

This extension inherits and adapts security considerations from:

- CAA framework [RFC8659],
- CAA Record Extensions for Account URI and (ACME) Binding [RFC8657],

- Certificate Transparency Version 2.0. [RFC9162],
- ACME protocol [RFC8555],
- SMTP MTA Strict Transport Security (MTA-STS) [RFC8461].

As an extension of the DNS-based CAA framework, this proposal generally falls within the Internet threat model defined by [RFC3552].

6.1. Global considerations

This specification augments the CAA RR framework by introducing finer-grained controls over both Certificate issuance and CT-Log submission. Domain owners can now explicitly authorize which CAs may issue Certificates and which CT-Log operators must or may record those Certificates. When used in tandem with compliant CAs and CT-Log operators, **"issuect"** significantly strengthens the issuance security posture for domains that opt in.

Using *"critical=true"* ensures that no Certificate is ever issued without logging to the specified CT-Log endpoints, preventing accidental or malicious bypass of transparency requirements.

By co-locating CT-Log authorization data within DNSSEC-protected CAA RRs, this extension also reduces reliance on out-of-band lists and third-party dependencies, thereby minimizing attack surface and improving resilience against CT-Log Operator compromise.

To further harden CAA **"issuect"** Policy discovery and protect against DNS manipulation, CAs **SHOULD** implement a CAA-CT-STS HTTPS fallback mechanism. This provides a second, TLS-authenticated channel for retrieving CAA **"issuect"** Policies directly from the domain owner.

6.1.1. CAs Preferred Proposed Validation Path

The **RECOMMENDED** multi-stage retrieval process is:

1. **DNSSEC-first:** Attempt to fetch the CAA **"issuect"** RR via DNS, validating all responses with DNSSEC. If the RR validates successfully, process it per normal CAA logic.
2. **Well-Known HTTPS (fast-path):** If DNSSEC validation fails, or the zone is not signed, perform a single HTTPS GET to `"https://caa-ct-sts.<domain>/.well-known/caa-ct-sts.txt"`, resolving the hostname only to its A/AAAA address (no further name recursion), to minimize exposure to poisoned or spoofed DNS replies.
3. **IP-only HTTPS (slow-path):** If step 2 fails, explicitly fetch the A/AAAA record for `"caa-ct-sts.<domain>"`, then open a TLS connection by IP address (SNI still `"caa-ct-sts.<domain>"`) to retrieve the same policy file.
4. **Insecure DNS with ODoH:** Should all prior steps be unreachable, fall back to fetching CAA **"issuect"** RRs over standard DNS, but prefer an Oblivious DoH (ODoH) [RFC9230] transport to obscure queries from on-path attackers.

By layering these channels, DNSSEC, well-known HTTPS, IP-only HTTPS, and finally ODoH-protected DNS, the CA dramatically reduces the attack surface for policy tampering, while still providing robust fallback when certain transports are unavailable.

6.2. CAA-CT-STS Advantages and Disadvantages

Advantages: The HTTPS source prevents a DNS attacker from simply replacing or deleting the DNS CAA RR Extension "**issuect**", they would also have to manipulate the HTTPS connection. If the DNS (CA side) fails, for example, the CA can use this defined backup channel to determine which Logs to use. Since web servers are usually TLS-enabled anyway, the hurdle for domain owners would be relatively low.

Disadvantages: The domain owner needs a valid web certificate, which can initially be a "chicken and egg" problem (how does one provide `/.well-known/caa-ct-sts.txt` before one has a Certificate?). In addition, the CA must be configured to perform HTTPS retrieval in the first place, this is not provided for in the current CA/ACME implementations. Another point is that an attacker who is already performing MITM at the network level could manipulate both DNS and HTTPS, (e.g., via a fake Root Certificate), so that even the `/.well-known/caa-ct-sts.txt` approach only provides limited protection. MTA-STS solves this through HSTS and policy caching, among other things. Overall, HTTPS `/.well-known/caa-ct-sts.txt` would therefore be a useful additional measure that could serve as a secondary source of trust, especially in the absence of DNSSEC.

6.3. Policy Redundancy Considerations $|C| \geq n + 1 \wedge |W| \leq 2$

Let C be the number of critical CT-Logs and W be the number of whitelisted (non-critical) CT-Logs, then the following expression is strongly **RECOMMENDED**: $|C| \geq n + 1 \wedge |W| \leq 2$

While the "critical=true" flag in the CAA "**issuect**" Parameter enforces that every Certificate issuance must be logged to all specified CT-Logs, this strict requirement can introduce availability risks: if any one of those CT-Log endpoints becomes temporarily unreachable (maintenance, network partition, DDoS, etc.), the CA, per specification, must reject the Certificate Signing Request. To balance tamper-resistance against operational resilience, it is therefore strongly **RECOMMENDED** the following guardrails:

1. Minimum Redundancy (" $n + 1$ "). Let n be the minimum number of distinct, trusted CT-Logs mandated by major CT-Truststore policies (for instance, Apple and Google currently require a Certificate to be embedded with at least " $n = 2$ " or " $n = 3$ " unique SCTs from independent logs to be accepted in their roots). Domain owners **MUST NOT** specify less than $n + 1$ CT-Logs under "critical=true". This ensures that even if one of the critical CT logs is unavailable, the CA can still obtain the remaining n SCTs and proceed with issuance.
2. "+ 2" Whitelist of Non-Critical CT-Logs. In addition to the $n + 1$ critical logs, domain owners **SHOULD** nominate at least up to two further CT-Logs without the "critical=true" flag. These "whitelisted" CT-Logs provide extra transparency channels, enabling issuance to continue if a critical CT-Log fails, but do not block issuance if they are unreachable. They **MUST NOT** carry "critical=true"; otherwise, a transient failure at any one CT-Log would force a full issuance rejection. These non-critical logs help broaden auditing coverage (e.g., corporate or private logs) without jeopardizing certificate availability.

6.3.1. Operational Procedure for CAs

- Fetch and Validate CAA "**issuect**": Retrieve the CAA "**issuect**" RR (via DNSSEC or the HTTPS fallback) and parse the list of critical versus non-critical CT-Logs.
- Attempt Precertificate Submission: For each critical CT-Log, send the Precertificate and collect its SCT. If all critical logs successfully return SCTs, proceed. If any critical CT-Log fails, but at least n SCTs from the remaining critical CT-Logs are still obtained (thanks to the $n + 1$ redundancy), proceed; otherwise, reject the CSR.
- Optionally Submit to Non-Critical CT-Logs: Attempt to send it to non-critical CT-Logs in parallel. If they also fail or are slow, reject issuance; report any failures for domain-owner monitoring.

By embedding $n + 1$ redundancy and a $a + 2$ non-critical whitelist, this framework simultaneously enforces strong, mandatory CT logging (protecting against malicious or erroneous omission) and maintains the operational flexibility necessary to keep issuance flowing even during intermittent CT-Log outages.

6.4. Scope Limited to CAs and CT-Log Operators Processing CAA RRs

This extension empowers domain owners, who already maintain relationships with compliant CAs and CT-Log operators, to impose additional constraints on both Certificate issuance and log submission, provided those parties implement the "**issuect**" mechanism. It does not strengthen guarantees for any CA or CT-Log operator that does not support this extension: non-conforming entities retain their existing ability to issue Certificates or submit log entries regardless of "**issuect**" policy. Consequently, domains that authorize only participating CAs and CT-Log operators remain vulnerable to unauthorized issuance or logging by entities that ignore or merely advise upon CAA RRs. Even in DNSSEC-protected zones, the risk of misissuance persists if a CA or CT-Log operator fails to validate CAA RRs via a trusted, DNSSEC-validating resolver.

6.5. Additional CAA-CT-STS Security Considerations

When a domain owner already employs DNSSEC, it is **RECOMMENDED**, that the HTTPS endpoint serving the CAA-CT-STS Policy be further bound via DANE/TLSA records. By publishing TLSA RR for both leaf and intermediate certificates, CAs retrieving `"/.well-known/caa-ct-sts.txt"` gain an additional multipath validation and out-of-band binding to the X.509 chain:

1. TLSA Usage

- 3 1 1 — SHA-256 hash of the leaf certificate's SPKI
- 3 1 2 — SHA-512 hash of the leaf certificate's SPKI
- 2 1 1 — SHA-256 hash of the issuing intermediate certificate's SPKI
- 2 1 2 — SHA-512 hash of the issuing intermediate certificate's SPKI

Here, TLSA-usage 3 (DANE-EE) and 2 (DANE-TA), selector 1 (SPKI), and matching types 1 (SHA-256) and 2 (SHA-512) ensure that CAs validate the exact certificates used by the policy host, preventing MITM.

2. SVCB / HTTPS Record

One can define a SVCB (or HTTPS) DNS RR for "caa-ct-sts.<domain>", advertising support for HTTP/3 (QUIC) and pointing to the same IP addresses as the A/AAAA records. This allows CAs to connect over the most modern, UDP-based transport with built-in encryption and connection resilience.

3. Strict TLS Configuration

The policy server should only accept TLS1.3 with strong cipher suites (e.g., AES_256_GCM) and elliptic-curve key exchange using X448, P-521, or P-384 (in this order). This ensures forward secrecy and resistance to known TLS attacks.

4. Optional DANE-Style Binding in the Policy File (NOT implemented in this draft)

For maximum end-to-end integrity, the "/.well-known/caa-ct-sts.txt" file itself may include DANE-like fields to bind a specific DNSSEC KZK. For example:

```
version: CAACTSTSV 1
max_age: 6 0
kzk_id: 3 2 7 6 8
kzk_hash: 'MII(...)'
ct_policy: ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameNet 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.example.net/logs/eu/funnynamenet 2 0 2 5; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )
```

Figure 7: CAA-CT-STs Policy incl. DNSSEC KZK-Binding

Whereas "kzk_id" would be an integer identifier of the Key-Id of the KZK of the domain, and "kzk_hash" would be a Base64-encoded hash of the KZK of the domain. By combining DNSSEC-protected TLSA and in-CAA-CT-STs-Policy kzk_hash and kzk_id values, CAs gain cryptographic assurance of both the CAA-CT-STs Policy files authenticity and the CT-Log endpoints identity, further reducing the risk of policy-fetch or logging interception attacks.

6.6. Authorization Freshness

CAA allows a Certification Authority to pre-authorize an account, to request Certificates for a domain well before the actual issuance event. As a result, the CAA policy in effect at authorization time may differ from the policy published at issuance time. To mitigate this risk, CAs SHOULD adopt one or more of the following practices:

- Issue pre-authorizations with short validity intervals (for example, one hour).
- Re-validate the domain's current CAA policy immediately prior to certificate issuance.

By limiting the window during which stale authorizations remain valid and by re-checking CAA RRs at issuance time, CAs reduce the likelihood of complying with outdated or revoked policy statements.

6.7. Use with and without DNSSEC

The standard domain-validation model does not protect against an adversary capable of intercepting all traffic to a domain (a "global man-in-the-middle" [MITM] attack). Such an attacker can hijack validation requests from any CA and thus impersonate domain control. When a domain is DNSSEC-signed, and the CA resolves DNS exclusively via a trusted DNSSEC-validating resolver, however, the authenticity and integrity of DNS data are assured.

To leverage this protection, a CA's validation process **SHOULD** either:

- Rely solely on DNSSEC-authenticated DNS responses for all validation data; or
- Cryptographically bind every other validation channel (e.g., HTTP-01, TLS-ALPN-01) to material obtained via DNSSEC.

6.8. Scenario Effects for "issuect" and Security

DNSSEC-secured

- CAA and **"issuect"** RRs are authenticated.
- Any modification to these RRs is detected by DNSSEC validation failures.
- The CA can trust that CT-Log authorizations originate from the domain owner.
- Certificates cannot be issued "out of band," nor can CAA-driven CT-Log instructions be tampered with undetected.

Not DNSSEC-secured

- CAA and **"issuect"** RRs may be forged or suppressed.
- A CA relying on cached or unauthenticated responses risks seeing incorrect policies.
- An attacker could inject a false empty **"issuect"** ";" to disable logging, or point to malicious CT-Logs, subverting transparency.
- In extreme "SOA interception" scenarios, an attacker could entirely block certificate issuance by dropping CAA/A RRs.

6.9. Restrictions Supersedable by DNS Delegation

CAA RRs are discovered during validation by traversing up the DNS hierarchy until one or more RRs are located. Because of this lookup behavior, CAA RRs cannot reliably restrict or control Certificate issuance for subdomains whose DNS management has been delegated to another party (for example, via NS delegation or by granting limited DNS-editing rights for a subdomain).

7. References

7.1. Normative References

- [ISO-8601] International Organization for Standardization, "Date and time — Representations for information interchange", ISO ISO 8601:2019, June 2019, <<https://www.iso.org/standard/70907.html>>.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [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>>.
- [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>>.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <<https://www.rfc-editor.org/info/rfc2181>>.
- [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>>.
- [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>>.
- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <<https://www.rfc-editor.org/info/rfc5155>>.
- [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>>.
- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/info/rfc5234>>.
- [RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <<https://www.rfc-editor.org/info/rfc6234>>.

-
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <<https://www.rfc-editor.org/info/rfc6962>>.
- [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <<https://www.rfc-editor.org/info/rfc7234>>.
- [RFC7405] Kyzivat, P., "Case-Sensitive String Support in ABNF", RFC 7405, DOI 10.17487/RFC7405, December 2014, <<https://www.rfc-editor.org/info/rfc7405>>.
- [RFC7671] Dukhovni, V. and W. Hardaker, "The DNS-Based Authentication of Named Entities (DANE) Protocol: Updates and Operational Guidance", RFC 7671, DOI 10.17487/RFC7671, October 2015, <<https://www.rfc-editor.org/info/rfc7671>>.
- [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>>.
- [RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <<https://www.rfc-editor.org/info/rfc8551>>.
- [RFC8657] Landau, H., "Certification Authority Authorization (CAA) Record Extensions for Account URI and Automatic Certificate Management Environment (ACME) Method Binding", RFC 8657, DOI 10.17487/RFC8657, November 2019, <<https://www.rfc-editor.org/info/rfc8657>>.
- [RFC8659] Hallam-Baker, P., Stradling, R., and J. Hoffman-Andrews, "DNS Certification Authority Authorization (CAA) Resource Record", RFC 8659, DOI 10.17487/RFC8659, November 2019, <<https://www.rfc-editor.org/info/rfc8659>>.
- [RFC9364] Hoffman, P., "DNS Security Extensions (DNSSEC)", BCP 237, RFC 9364, DOI 10.17487/RFC9364, February 2023, <<https://www.rfc-editor.org/info/rfc9364>>.
- [RFC9495] Bonnell, C., "Certification Authority Authorization (CAA) Processing for Email Addresses", RFC 9495, DOI 10.17487/RFC9495, October 2023, <<https://www.rfc-editor.org/info/rfc9495>>.
- [RFC9499] Hoffman, P. and K. Fujiwara, "DNS Terminology", BCP 219, RFC 9499, DOI 10.17487/RFC9499, March 2024, <<https://www.rfc-editor.org/info/rfc9499>>.
- [X.680] International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), "Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T X.680 (06/2021), June 2021, <<https://www.itu.int/rec/T-REC-X.680-X.693-202102-I/en>>.
-

- [X.690] International Telecommunication Union, Telecommunication Standardization Sector (ITU-T), "Information technology – ASN.1 encoding rules: Specification of Basic, Canonical, and Distinguished Encoding Rules", ITU-T X.690 (07/2021), July 2021, <<https://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf>>.

7.2. Informative References

- [POSIX] The Institute of Electrical and Electronics Engineers; The Open Group, "Portable Operating System Interface (POSIX™) – Base Specifications", IEEE Std 1003.1-2017, The Open Group Base Specifications Issue 7, 2018 Edition, ISO/IEC 9945:2009/Cor 2:2017(E), December 2017, <<https://pubs.opengroup.org/onlinepubs/9699919799.2018edition/>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [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>>.
- [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>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<https://www.rfc-editor.org/info/rfc4035>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [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>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

- [RFC8461] Margolis, D., Risher, M., Ramakrishnan, B., Brotman, A., and J. Jones, "SMTP MTA Strict Transport Security (MTA-STS)", RFC 8461, DOI 10.17487/RFC8461, September 2018, <<https://www.rfc-editor.org/info/rfc8461>>.
- [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>>.
- [RFC9162] Laurie, B., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <<https://www.rfc-editor.org/info/rfc9162>>.
- [RFC9230] Kinnear, E., McManus, P., Pauly, T., Verma, T., and C.A. Wood, "Oblivious DNS over HTTPS", RFC 9230, DOI 10.17487/RFC9230, June 2022, <<https://www.rfc-editor.org/info/rfc9230>>.

7.3. URI

- [URI1] "This document", <<https://coresecret.dev/msw/draft-weidner-catalog-rr-ext.git>>.
- [URI2] "7A83 41E5 F157 0319 D80F 4418 A11E 8851 9A3D 8CF6", <<https://keys.openpgp.org/vks/v1/by-fingerprint/7A8341E5F1570319D80F4418A11E88519A3D8CF6>>.
- [URI3] "Known Logs", <<https://github.com/google/certificate-transparency-community-site/blob/master/docs/google/known-logs.md>>.
- [URI4] "JSON List of CT-Logs that are currently compliant with Chromes CT policy", <https://www.gstatic.com/ct/log_list/v3/log_list.json>.
- [URI5] "Apple's Certificate Transparency policy", <<https://support.apple.com/en-us/103214>>.
- [URI6] "JSON List of CT-Logs that are currently compliant with Apple's CT policy", <https://valid.apple.com/ct/log_list/current_log_list.json>.
- [URI7] "Chrome's Certificate Transparency Policy", <https://googlechrome.github.io/CertificateTransparency/ct_policy.html>.
- [URI8] "Public Key Infrastructure using X.509 (PKIX) Parameters", <<https://www.iana.org/assignments/pkix-parameters/pkix-parameters.xhtml>>.
- [URI9] "Well-Known URIs", <<https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml>>.

Appendix A. Informational: CAA "issuect" Property Tag Processing Checklist

(1)

Discover Relevant RRSet

- Perform DNS lookup for the domain's CAA RRSet using the algorithm from [Section 3 of \[RFC8659\]](#)

(2) Check for "issuect" Presence

- If the RRSet is empty or contains no **"issuect"** RRs, proceed without CT-Log restrictions.

(3) Repeat for Each Domain

- For multi-domain Certificates (e.g., via SANs), repeat steps 1–2 for each domain name.

(4) Validate issue / issuewild Authorization

- Independently verify any "issue" or "issuewild" CAA RRs before considering CT-Log restrictions.
- If CAA authorization and **"issuect"** parameters reference different CAs, treat the configuration as unsatisfiable.

(5) Parse and Validate issuect Parameters

- Ensure exactly eight parameters appear in the correct order (see Section 3.3).
- Enforce RFC 2119 rules:
 - critical: must be "true" or "false".
 - desc: single-quoted, no internal '.
 - validfrom / validtill: strict YYYY-MM-DDThh:mm:ssZ.
 - cturi: absolute URI with lowercase hostname.
 - logid / pubkey: single-quoted Base64 values.
- Trailing semicolon only after pubkey.
- Treat any formatting errors as if the RR does not exist.

(6) Enforce Whitelist and Default-Deny Semantics

- If one or more valid **"issuect"** RRs exist:
- Permit CT-Log submissions only to the union of all whitelisted URIs.
- If any RR has "critical=true", reject issuance unless all issued SCTs are logged to a whitelisted CT-Log.
- If an empty **"issuect" ";"** RR is present, and no non-empty RRs exist, prohibit all CT-Log submissions.

(7) Determine Required SCT Count

- For each CA, obtain at least $n + 1$ unique CAA **"issuect"** RRs, where n is the minimum SCT count (Section 4.1).

- Consult CA policies (e.g., Google's or Apple's CT-Log requirements) for the exact value of n .

(8) **Proceed with Certificate Issuance**

- Only after all the above checks pass, CAA authorization, "**issuect**" validation, SCT count and logging requirements, issue the Certificate.

Appendix B. Informational: DNS Zone-file Examples

B.1. Preface

The following examples are provided for informational purposes only and are supplied "as is", without a warranty of any kind or liability. They illustrate relevant RRSets, and their expected processing semantics. Their BIND file notation format is defined in [\[RFC1035\]](#). Parentheses are used to ignore a line break in DNS zone-file presentation format, as per [Section 5.1](#) of [\[RFC1035\]](#).

B.2. CAA "issuect" Comprehensive and Proper Example

The following shows a comprehensive and proper example DNS zone-file fragment of the "example.com" zone that nominates six distinguished CT-Log URIs for the "example.ca" Certification Authority, which must ($n + 1$) and could ($+ 2$) [redundancy rule [Section 6.3](#)], log any Certificates requested to the specified CT-Log, as authorized to log >180-day valid Certificates for the "sub.example.com" domain. The submission is limited to the critical CT-Logs "ct.example.net", "ct.example.org", "ct.example.sh", and "ct.example.xyz" and to the non-critical CT-Logs "ct.non-critical.net" and "ct.non-critical.org".

NOTE: "CAA authorizations are additive; thus, the result of specifying both the empty issuer, and a specified issuer is the same as specifying just the specified issuer alone."


```
sub.example.com. 6 0 IN CAA 0 issuet ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameNet 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.example.net/logs/eu/funnynamenet 2 0 2 5; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )

sub.example.com. 6 0 IN CAA 0 issuet ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameOrg 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.example.org/logs/eu/funnynameorg 2 0 2 5"; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )

sub.example.com. 6 0 IN CAA 0 issuet ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameSh 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.example.sh/logs/eu/funnynamesh 2 0 2 5; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )

sub.example.com. 6 0 IN CAA 0 issuet ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameXyz 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.example.xyz/logs/eu/funnynamexyz 2 0 2 5; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )

sub.example.com. 6 0 IN CAA 0 issuet ( "example.ca; \
critical=false; \
desc='Example Log NonCriticalNet 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.non-critical.net/logs/eu/noncriticalnet 2 0 2 5; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )

sub.example.com. 6 0 IN CAA 0 issuet ( "example.ca; \
critical=false; \
desc='Example Log NonCriticalOrg 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.non-critical.org/logs/eu/noncriticalorg 2 0 2 5; \
logid='sh 4 (...)'; \
```

```
pubkey='MFk(...);' )
```

Figure 8: Zone-file: CAA "issuect" Comprehensive and Proper Example

B.3. CAA "issuect" No CT-Log Submission Example

The following shows a no CT-Log submission at all example DNS zone-file fragments of the "example.com" zone that wishes to not submit their Certificates to any CT-Log.

```
sub.example.com. 6 0 IN CAA 0 issuect ";"
```

Figure 9: Zone-file: CAA "issuect" No CT-Log Submission Example

B.4. CAA "issuect" Erroneous Example

The following shows an erroneous example DNS zone-file fragment of the "example.com" zone that falsely set the Parameter Set, so that according to the Processing definitions, the erroneous Property will be treated as non-existent.

```
sub.example.com. 6 0 IN CAA 0 issuect ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameXyz 2 0 2 5'; \
validfrom=; \
validtill=; \
cturi=https://ct.example.xyz/logs/eu/funnynamexyz 2 0 2 5; \
logid='sh 4 (...)'; \
pubkey='MFk(...);' )
```

Figure 10: Zone-file: CAA "issuect" Erroneous Example

B.5. CAA-CT-STs "_caa-ct-sts" Example

```
_caa-ct-sts.example.com. 6 0 IN TXT "v=CAACTSTsv 1 ; id= 2 0 2 5 0 4 2 7 1 2 1 2 1 2 Z;"
```

Figure 11: Zone-file: CAA-CT-STs "_caa-ct-sts" Example

Appendix C. Informational: CAA-CT-STS Examples

C.1. CAA-CT-STS Policy File Example

The following example indicates that Certificates for the domain must be logged in:

- <https://ct.example.net/logs/eu/funnynamenet2025>
- <https://ct.example.org/logs/eu/funnynameorg2025>
- <https://ct.example.sh/logs/eu/funnynamesh2025>
- <https://ct.example.xyz/logs/eu/funnynamexyz2025>

```
version: CAACTSTSV 1
max_age: 6 0
ct_policy: ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameNet 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.example.net/logs/eu/funnynamenet 2 0 2 5; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )

ct_policy: ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameOrg 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.example.org/logs/eu/funnynameorg 2 0 2 5"; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )

ct_policy: ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameSh 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.example.sh/logs/eu/funnynamesh 2 0 2 5; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )

ct_policy: ( "example.ca; \
critical=true; \
desc='Example Log FunnyNameXyz 2 0 2 5'; \
validfrom= 2 0 2 5 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
validtill= 2 0 2 6 - 0 1 - 0 1 T 0 0 : 0 0 : 0 0 Z; \
cturi=https://ct.example.xyz/logs/eu/funnynamexyz 2 0 2 5; \
logid='sh 4 (...)'; \
pubkey='MFk(...)';" )
```

Figure 12: CAA-CT-STs Policy File Example

Appendix D. Informational: POSIX compliant Scripts

The following Shell scripts are provided for informational purposes only and are supplied "as is", without a warranty of any kind or liability.

D.1. Shell "issuect" Generator script

The following POSIX [\[POSIX\]](#)-compliant shell script retrieves information from the regularly updated Google curated CT-Log-Operators file [JSON List of CT-Logs that are currently compliant with Chromes CT policy \[4\]](#) and generates zone-file compliant DNS RRs.


```

# !/bin/sh
set -eu
readonly OWN_DOMAIN="$ 1 "
readonly CAA_DOMAIN="$ 2 "
readonly CRIT_FLAG="$ 3 "
readonly ZONE_FILE="zone_${OWN_DOMAIN}_CAA.txt"
case "${CRIT_FLAG}" in
  true|false) ;;
  * ) echo "Error: CRIT_FLAG MUST be either 'true' or 'false'." >& 2
    exit 1
    ;;
esac
:> "${ZONE_FILE}"
JSON=$(curl -fsSL https://www.gstatic.com/ct/log_list/v 3 /log_list.json)
readonly JSON
echo "${JSON}" | awk -v OWN="${OWN_DOMAIN}" -v CA="${CAA_DOMAIN}" -v CRIT="$
{CRIT_FLAG}" -v OUT="${ZONE_FILE}" '
BEGIN { FS="\""; }
/[[[:space:]] * "description"/ {
  desc=""; url=""; start=""; endt=""; logid=""; key="";
}
/"description"/ {
  desc = $ 4
  gsub(/\\ 0 4 7 /, "", desc)
}
/"url"/ {
  url = $ 4
}
/"start_inclusive"/ {
  start = $ 4
}
/"end_exclusive"/ {
  endt = $ 4
}
/"log_id"/ {
  logid = $ 4
}
/"key"/ {
  key = $ 4
  gsub(/\\ 0 4 7 /, "", key)
}
/"end_exclusive"/ {
  if (desc != "" && url != "" && start != "" && logid != "" && key != "") {
    printf "%s. 6 0 IN CAA 0 issuect ( \"%s; critical=%s; desc=\"%s\"; validfrom=%s; validtill=%s;
cturi=%s; logid=\"%s\"; pubkey=\"%s\";\\\" )\\n\", \\
      OWN, CA, CRIT, desc, start, endt, url, logid, key \\
    >> OUT
  }
}

```

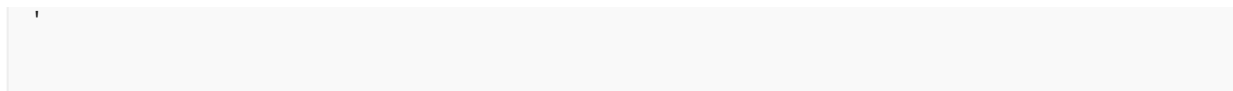


Figure 13: Shell "issuect" Generator script

D.2. Shell "caa-ct-sts.txt" Generator script

The following POSIX [\[POSIX\]](#)-compliant shell script retrieves information from the regularly updated Google curated CT-Log-Operators file [JSON List of CT-Logs that are currently compliant with Chromes CT policy \[4\]](#) and generates a "caa-ct-sts.txt"-Policy file.

```

#!/bin/sh
set -eu
readonly OWN_DOMAIN="$ 1 "
readonly CAA_DOMAIN="$ 2 "
readonly CRIT_FLAG="$ 3 "
readonly CAA_CTS_TS="caa-ct-sts.${OWN_DOMAIN}.txt"
case "${CRIT_FLAG}" in
  true|false) ;;
  * ) echo "Error: CRIT_FLAG MUST be either 'true' or 'false'." >& 2
    exit 1 ;;
esac
:> "${CAA_CTS_TS}"
{ echo "version: CAACTSTSV 1 "
  echo "max_age: 6 0 "
} > "${CAA_CTS_TS}"
JSON=$(curl -fsSL https://www.gstatic.com/ct/log_list/v 3 /log_list.json)
readonly JSON
echo "${JSON}" | awk -v OWN="${OWN_DOMAIN}" -v CA="${CAA_DOMAIN}" -v CRIT="$
{CRIT_FLAG}" -v OUT="${CAA_CTS_TS}" '
  BEGIN { FS="\""; }
  /[[[:space:]] * "description"/ {
    desc=""; url=""; start=""; endt=""; logid=""; key="";
  }
  /"description":/ {
    desc = $ 4
    gsub(/ 0 4 7 /, "", desc)
  }
  /"url":/ {
    url = $ 4
  }
  /"start_inclusive":/ {
    start = $ 4
  }
  /"end_exclusive":/ {
    endt = $ 4
  }
  /"log_id":/ {
    logid = $ 4
  }
  /"key":/ {
    key = $ 4
    gsub(/ 0 4 7 /, "", key)
  }
  /"end_exclusive":/ {
    if (desc != "" && url != "" && start != "" && logid != "" && key != "") {
      printf "ct_policy: ( \"%; critical=%s; desc=\"%s\"; validfrom=%s; validtill=%s; cturi=%s;
logid=\"%s\"; pubkey=\"%s\"; \" )\n", \
        CA, CRIT, desc, start, endt, url, logid, key \
      >> OUT
    }
  }
}

```




Figure 14: Shell "caa-ct-sts.txt" Generator script

Appendix E. Informational: DNS TTL

E.1. High-Security, High-Change Environments

It is **RECOMMENDED** to prefer very low TTL (60–300 s). In case of compromise or mistakes a real time reaction is still possible.

E.2. Stable, Low-Risk Domains with DDoS Concerns

It is **RECOMMENDED** to prefer high TTL (86 400 s+) to minimize DNS-load and reduce resolver queries.

E.3. Balanced Approach for Most

It is **RECOMMENDED** to use a medium TTL (e.g., 3 600 s / 1 h). It limits exposure to cache-poisoning and CA mis-issuance to an hour, while keeping query volume manageable.

Appendix F. Informational: Change History

F.1. draft-weidner-catalog-rr-ext-00

Initial Version

Acknowledgements

This Draft uses extracts from templates written by Pekka Savola, Elwyn Davies, and Henrik Levkowetz.

Contributors

The author gratefully acknowledges the contributors rigorous and professional critique, which materially improved the technical clarity and robustness of this draft. Their objective and insightful feedback has been invaluable in refining the specification.

André Horst Zimnol

Private Contributor

Email: rfc.editor@zimnol.net

Author's Address

Marc Simon Weidner (EDITOR)

Centurion Intelligence Consulting Agency

Phone: +1 (202) 992 1702

Email: rfc.editor@coresecret.eu

URI: <https://coresecret.eu/>