Discussion: The future of “technically constrained” subordinate CAs

Presented to the Server Certificate Working Group
at Face-to-Face 66 (October 15, 2025)

Background

- **Ballot 105 (2013)** added the definition of “Technically Constrained Subordinate CA Certificate” to the BRs:

  > “*A Subordinate CA certificate which uses a combination of Extended Key Usage settings and Name Constraint settings to limit the scope within which the Subordinate CA Certificate may issue Subscriber or additional Subordinate CA Certificates*”

Background

- **Ballot 105** also added a third-party audit exemption for these CAs:

> *"Certificates that are capable of being used to issue new certificates MUST either be Technically Constrained in line with section 9.7 and audited in line with section 17.9 only, or Unconstrained and fully audited in line with all remaining requirements from section 17."*

This exemption still exists in Section 8.1 ("Frequency or circumstances of assessment").

Background

- **Ballot 187 (2017)** introduced CAA record checking requirements and an exemption for doing so by technically constrained subordinate CAs:

  > "*CAA checking is optional for certificates issued by an Technically Constrained Subordinate CA Certificate as set out in Baseline Requirements section 7.1.5, where the lack of CAA checking is an explicit contractual provision in the contract with the Applicant.*"

  This exemption still exists in Section 3.2.2.8 ("CAA Records").

Background

- **Ballot SC-062 (2023)** was a multi-year effort aimed at better aligning certificate context expectations across issuers and consumers by offering a more precise certificate profile specification.

  Some of its intended benefits were to:
  - reduce the opportunity for confusion
  - promote more consistent and reliable implementations

- This ballot introduced the current text of **Sections 7.1.2.3 and 7.1.2.5**, now referenced by the current CAA exemption introduced on the previous slide.

Background

- **Section 7.1.2.3** is the non-TLS subordinate CA certificate profile, and should be considered out of scope.

- **Section 7.1.2.5:**
  - ...categorizes a CA being “technically constrained” by the presence of Name Constraints (RFC 5280).
  - ...vaguely points to 3.2.2.4 to describe how an Applicant has registered a permitted Subtree or has been assigned an ipAddress block.
  - ...does not clearly describe how CA certificate validity interplays with validation reuse periods.

Problem statement

- The construct of a “technically constrained subordinate CA” is manufactured and only known to this community.
- By treating “technically constrained” subordinate CAs differently than “ordinary” subordinate CAs, we are contributing to unnecessary ecosystem complexity.
- Allowing exemptions for technically constrained subordinate CAs should be seen as reducing transparency, weakening security assumptions, and creating opportunities for problems.
- The TLS BRs are misaligned with existing root store policies.

Use of Name Constrained CAs

- 12*/1719 (~.7%) of subordinate CA certificates trusted in Chrome have Name Constraints present.

  Note (*): 9 of these 12 contain the same public key.

- This represents:
  - 4 constrained CA public keys, belonging to 3 CA Owners.
  - ~150 leafs, or 0.000011% of unexpired precertificates known to our monitoring solution.

Recommendation

- Sunset the concept of technically constrained subordinate CAs and all related exemptions.

Additional opportunity

- Further simplify the TLS BR profiles and promote alignment with root store operator policies that expect purpose-specific PKI hierarchies.
    - Fewer profiles
    - Less or no inheritance across sections

What do we mean by “simplify”?

- We should sunset practices that have largely been abandoned, for example, IV certificates (where adoption appears very, very low), just as we did with the precertificate signing CA profile in SC-092.

- We should sunset practices that no longer make sense in the context of the public TLS ecosystem in the year 2025, for example, use of SHA-1.

- We should move “profile” related requirements to Section 7.
  - Why is it better to have certificate lifetime described in Section 6.3.2 (“Certificate operational periods and key pair usage periods”) and *referenced* by each profile than directly in each profile?

What could Section 7 look like?

- Section 7.1.2.1 CA Certificates
  - Section 7.1.2.1.1 - Root CA Certificate Profile
  - Section 7.1.2.1.2 - TLS Server Authentication Subordinate CA Certificate Profile
  - Section 7.1.2.1.3 - TLS Server Authentication and Client Authentication Subordinate CA Certificate Profile

- Section 7.1.2.2 - Subscriber (End-Entity) Certificates
  - Section 7.1.2.2.1 - Domain Validated Certificate Profile
  - Section 7.1.2.2.2 - Organization Validated Certificate Profile
  - Section 7.1.2.2.3 - Extended Validation Certificate Profile

- Section 7.1.2.3 Infrastructure Certificates
  - Section 7.1.2.3.1 - Precertificate Profile
  - Section 7.1.2.3.2 - OCSP Responder Certificate Profile

What do we mean by “less inheritance”?

- Today, an ecosystem participant needs to navigate across 10+ tables, scattered across multiple subsections, to “understand” the expected contents of a DV certificate.

- Rather than prioritizing simplicity, we’re prioritizing the avoidance of redundancy. **Is this working as intended?**

7.1.2.7 Subscriber (Server) Certificate Profile

Field	Description
tbsCertificate	
version	MUST be v3(2)
serialNumber	MUST be a non-sequential number greater than zero (0) and less than 2^{159} containing at least 64 bits of output from a CSPRNG.
signature	See Section 7.1.3.2
issuer	MUST be byte-for-byte identical to the subject field of the Issuing CA. See Section 7.1.4.1
validity	
notBefore	A value within 48 hours of the certificate signing operation.
notAfter	See Section 6.3.2
subject	See Section 7.1.2.7.1
subjectPublicKeyInfo	See Section 7.1.3.1
issuerUniqueID	MUST NOT be present
subjectUniqueID	MUST NOT be present
extensions	See Section 7.1.2.7.6
signatureAlgorithm	Encoded value MUST be byte-for-byte identical to the tbsCertificate.signature.
signature	

Discussion

Define Next Steps