



Further improving Section 7 of the TLS BRs

Presented to the Server Certificate Working Group
at Face-to-Face 68 (September 23, 2026)

Let's have a productive session!



- This session is about setting common direction.
- We should focus on root causes.
- Community participation is expected.

Candid expectations



- Our measure of success for this session is moving past familiar historical challenges to establish a definitive, clear path toward improvement.
 - We will leave this session with committed problem definitions and concrete ballot drafting tracks.
 - The result of that planned work will drive systemic improvement across the ecosystem and create new opportunities going forward.

Why improve profiles?



- **Advance ecosystem transparency:** Everyone knows what a CA intends to issue / is capable of issuing, rather than trying to reverse-engineer after the fact.
- **Promote simplicity:** Complexity leads to operational mistakes. Every optional field, conditional rule, or cross-referenced table (or policy) increases the opportunity for problems.
- **Eliminate avoidable misissuance:** Certificate profiles are deterministic structures. Improved profiles enable improved linting, and hopefully, can eliminate an avoidable class of failure mode.

This talk picks up from F2F 66



Discussion: The future of “technically constrained” subordinate CAs

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

- A focus on modernization resulted in [Ballot SC097: Sunset all remaining use of SHA-1 signatures in Certificates and CRLs](#)
- Recommended additional changes addressed in today’s presentation

CCADB Policy expectations



4. Policy Disclosures

CCADB enables associating PKI policy documents in the records for both root CA certificates and subordinate CA certificates.

CA Owners with either (1) a certificate included in a Root Store or (2) a CA certificate that validates to a certificate included in a Root Store MUST minimally:

- accurately describe the policies and practices of their CA(s) within a Certificate Policy (CP) and corresponding Certification Practice Statement (CPS), or preferably, a single document combined as a CP/CPS.
- ensure the CA policy documents are:
 - freely downloadable and publicly available for examination;
 - available in an authoritative English language version; and
 - sufficiently detailed to assess the operations of the CA(s) and compliance with the expectations set forth in this Policy, the applicable CA/Browser Forum Baseline Requirements, and any applicable Root Store Operator policies, and MUST NOT conflict with any of the requirements specified therein.

“...accurately describe practices...” ?



- **Actual practices vs. theoretical**

- A policy cannot merely restate the outer bounds of what the BRs permit and be considered “accurate.”
- Related to certificate profiles, a policy must describe what the CA *actually* does in its issuance pipeline.

- **Profiles are operational commitments**

- Members of the ecosystem rely on the (CP/)CPS as an authoritative specification.
- If a policy says certificates contain specific fields, that is a binding operational commitment - not an illustrative suggestion.

“...sufficiently detailed to assess the operations...”



- **Is the opposite of "compliance by ambiguity"**
 - High-level generalities prevent independent verification.
 - If an auditor, relying party, or community member cannot tell whether a specific certificate should or shouldn't have been issued, the policy fails this standard.
- **We need auditable granularity**
 - Detailed profiles give operational ground truth to evaluate against.
 - Vague language invites certificate problem reports, whereas precision protects operations.

Speaking of audits...



- An auditor can only measure conformance against an objective baseline - generic commitments to "follow the BRs" and/or "illustrative" profiles give them nothing specific to test.
- ISO 9000/9001 quality management and compliance auditing... "*Say what you do, do what you say, and prove it.*"
 - If you don't say what you do, how can anyone ever prove you did it?
- Yesterday, the WTTF update highlighted the importance of actual practices, **including profiles**, matching documented practices and commitments.

Pressure testing expectations



- Does a policy that only restates adherence to the BR floor meet expectations?
- Does a policy that only offers illustrative / non-binding profiles meet expectations?
- If non-conformance can be argued away through grammatical ambiguity, was the profile ever truly “*sufficiently detailed to assess operations*”?
- Do generic policies provide CAs sufficient guardrails to build effective pre-issuance linters?
- Have we established a standards framework that sets CAs, auditors, and community members up to succeed?

Pressure testing expectations

- Does a policy that only restates adherence to the BR floor meet expectations?
- Does a policy that only offers illustrative / non-binding profiles meet expectations?
- If non-conformance can be argued away through grammatical ambiguity, was the profile ever truly “*survient, dedicated to assuring operations*”?
- Do generic policies provide sufficient guardrails to build effective pre-issuance intent?
- Have we established a standards framework that sets CAs, auditors, and community members up to succeed?

No.

Profiles over time



- **Pre-SC-62:**
 - Prose-heavy, scattered rules across RFC 5280, BRs, and CA practice statements.
 - Incomplete and unconstrained interpretations.
- **SC-62:**
 - Multi-year effort introducing explicit tabular profiles.
 - Dramatically increased consistency across implementations.
 - Paved the way for modern linting.
- **Opportunity to Improve:**
 - Tabular inheritance across 10+ tables created cognitive and implementation overhead.
 - Avoiding textual redundancy was prioritized over operational simplicity for implementers.

Discuss today's challenges

Conversation starters

- **Scattered requirements:** Navigating across prose, tables, and sections (e.g., certificate lifetimes stranded in Section 6.3.2).
- **Optionality leads to complexity:** Every optional field or alternative encoding expands surface area for mistakes.
- **Natural language pitfalls:** English prose in technical standards leads to diverging interpretations and semantic disputes.
- **The human cost:** Humans make mistakes, and in this ecosystem, that can bring widespread disruption.
- **We need better tools:** Automate away human risk.

We can make this better!



- <https://github.com/ryancdickson/staging/blob/improve-profiles-consolidated>
- **Summary of in-scope of changes:**
 - A. Improve profile disclosure & logging expectations
 - B. Strengthen linting expectations
 - C. Restructure profiles and create “standalone” tables
 - CA Certificate Profiles
 - TLS Subscriber (End-Entity) Certificate Profiles
 - Infrastructure Certificate Profiles (i.e., OCSP)
 - D. Sunset several profiles and related exemptions
 - IV certificates
 - Cross-certificates (instead treat these as subordinate CA certificates)
 - Anything with “Technically-Constrained” in the title

A few points worth mentioning...



- The branch imagines a destination. We can decide together if/how we get there.
 - Perhaps it's an omnibus ballot with a “far-off” effective date, perhaps it's 3 smaller ballots.
- For the *most part*, this update aimed for parity with today's BRs.
 - We'll discuss some intentional divergences and potential impacts.
- It is not ready to move forward as-is, but would benefit from feedback.
- It could go further on removing optionality and complexity.
- Iteration is expected.
 - The consolidated profile tables introduce a new format. Personal preferences will vary.

Unlocking future opportunity...



- Improved profile disclosures may pave the way for reimagining PKI Policy documents. For example...
 - **From:** CP + CPS (or combined CP/CPS)
 - **To:** [Issuance Practice Statement](#) (IPS) + Profiles
 - IPS being analogous to a survey / “Mad Libs” compared to today’s 100+ page documents.
- Related to Lucy’s talk yesterday, improved disclosures may also unlock additional opportunities with AI to complement (not replace) existing CA Owner compliance regimens.

Change summary and potential impacts

A. Improve profile disclosure & logging expectations



- Update Section 7.1 (“Certificate profile”):
 - Codify the expectation that Certificates strictly conform to disclosed, “default deny” certificate profiles.
 - Establish an expectation that CA Owners disclose an accurate public mapping of each Issuing CA to its permitted Profile(s).
 - Recommend CA Owners generate disclosed profiles directly from active CA system configurations to prevent manual transposition errors.
 - This must become a “must.”
- Update Section 5.4.1 (“Types of events recorded”):
 - Establish an expectation that CA and Subscriber Certificate issuance log entries record the specific Certificate Profile used to issue each Certificate.

A. Improve profile disclosure & logging ... [cont]



- Why (perceived ecosystem benefits):
 - Shifts CP/CPS profiles from illustrative to binding, deterministic schemas.
 - Coupling Section 5.4.1 to unique Profile Identifiers gives auditors an unambiguous join key between issued certificates, CA logs, and the CP/CPS.
 - Granular, “default deny” profile disclosures empower CAs to deploy profile-specific linters that block misissuance before certificates leave the issuance pipeline.
 - Generating disclosed profiles directly from active CA system configurations ensures published CP/CPS commitments match what the CA software actually enforces.

B. Strengthen linting expectations



- Update 4.3.1.2 (“Linting of to-be-signed Certificate content”):
 - Phased (“SHOULD” -> “MUST”) implementation where each to-be-signed artifact conforms to the specific Certificate Profile it was issued against.
- Update 4.3.1.3 (“Linting of issued Certificates”):
 - Effective a future date, the CA SHOULD use an automated Linting process to test 100% of issued Certificates for conformity with their applicable disclosed Certificate Profile *before* delivery to subscriber.
- Update 8.7 (“Self-Audits”)
 - Effective a future date, the CA's quarterly self-audit SHALL include automated Linting of 100% of Certificates issued during the quarter (*not a 3% sample*) to verify conformity against both the BRs and the applicable Certificate Profile.

B. Strengthen linting expectations [cont]



- Why (perceived ecosystem benefits):
 - Systematically eliminates major sources of certificate misissuance. By enforcing the CA's own default-deny CP/CPS profile rules directly in the issuance pipeline, unexpected fields, extensions, and profile drift are prevented before a certificate is ever signed with a trusted key.
 - Upgrading quarterly self-audit linting from a 3% sample to 100% of issued certificates ensures every issued certificate is (eventually) verified against the CA's CP/CPS profile.
 - A phased approach reduces friction and allows fine-tuning before linting failures can block issuance.

C. Consolidate profile tables



- Update [Section 7.1](#) (“Certificate profile”) & [Section 7.2](#) (“CRL profile”):
 - Consolidates multiple sections and tables into 7 standalone tables:
 - **CA Certificates (7.1.2.1):** Self-Signed Root CA (7.1.2.1.1), Server Auth Sub CA (7.1.2.1.2), Server + Client Auth Sub CA (7.1.2.1.3)
 - **TLS Subscriber Certificates (7.1.2.2):** DV (7.1.2.2.1), OV (7.1.2.2.2), EV (7.1.2.2.3)
 - **Infrastructure Certificates (7.1.2.3):** OCSP Responder Certificate Profile (7.1.2.3.1)
 - Standardized 5-column “blueprint” that specifies "Field / Extension", "OID", "Presence", "Criticality", and "Content & Validation Rules"
 - Replace 4 CRL tables with a single 5-column CRL Profile table.

C. Consolidate profile tables [cont]



- Why (perceived ecosystem benefits):
 - Community members no longer need to mentally join 18 scattered subsections (e.g., 7.1.2.7, 7.1.2.7.6–12, 7.1.2.11, 7.1.4.2, 6.3.2) to determine what is permitted in a DV, OV, or EV certificate.
 - Consolidating into self-contained tables preserves normative requirements while making profiles immediately readable top-to-bottom.
 - Gives CA Owners a clear, standardized single-table blueprint they can mirror directly in Section 7.1 of their own CP/CPS.
 - It's worth discussing how that should be done!

Field / Extension	OID	Presence	Criticality	Content & Validation Rules
<code>tlsCertificate.version</code>	—	MUST	—	Integer value 2 (v3).
<code>tlsCertificate.serialNumber</code>	—	MUST	—	Non-sequential positive INTEGER greater than zero (0) and less than 2 ³⁰ containing at least 64 bits of output from a CSPRNG.
<code>tlsCertificate.signature</code>	—	MUST	—	Permitted CA signature AlgorithmIdentifier matching outer signatureAlgorithm byte-for-byte.
<code>L. RSASSA-PKCS1-v1_5</code>	1.2.848.113549.1.1.11 / .12 / .13	Conditional	—	Explicit NULL parameters (0500). Exact DER hex encoding: • SHA-256 (1.2.848.113549.1.1.11): 300906892a864886f78001010500a11c301a06092a064806f780010108100806090686480165830402010500a203020130 • SHA-384 (1.2.848.113549.1.1.12): 300906892a864886f78001010500a11c301a06092a064806f780010108100806090686480165830402010500a203020130 • SHA-512 (1.2.848.113549.1.1.13): 300906892a864886f78001010500a11c301a06092a064806f780010108100806090686480165830402010500a203020130
<code>L. RSASSA-PSS</code>	1.2.848.113549.1.1.10	Conditional	—	Exact DER hex encoding: • SHA-256, MGF-1 w/ SHA-256, 32B salt: 304106092a064806f78001010500a11c301a06090686480165830402010500a11c301a06092a064806f780010108100806090686480165830402010500a203020130 • SHA-384, MGF-1 w/ SHA-384, 48B salt: 304106092a064806f78001010500a11c301a06090686480165830402010500a11c301a06092a064806f780010108100806090686480165830402010500a203020130 • SHA-512, MGF-1 w/ SHA-512, 64B salt: 304106092a064806f78001010500a11c301a06090686480165830402010500a11c301a06092a064806f780010108100806090686480165830402010500a203020130
<code>L. ECDSA</code>	1.2.848.10045.4.3.2 / .3 / .4	Conditional	—	Hash algorithm MUST match signing key curve. Exact DER hex encoding: • P-256 w/ SHA-256 (1.2.848.10045.4.3.2): 300a06082a0648cc30040302 • P-384 w/ SHA-384 (1.2.848.10045.4.3.3): 300a06082a0648cc30040303 • P-521 w/ SHA-512 (1.2.848.10045.4.3.4): 300a06082a0648cc30040304
<code>tlsCertificate.issuer</code>	—	MUST	—	Encoded value MUST be byte-for-byte identical to the encoded subject of the Issuing CA certificate.
<code>tlsCertificate.validity</code>	—	MUST	—	Encoded as <code>UTCTime</code> (dates through 2049) or <code>GeneralizedTime</code> (dates 2050+). • <code>notBefore</code> : Value MUST be no earlier than 48 hours prior to signing and no later than the time of signing. • <code>notAfter</code> : Certificate validity period MUST NOT exceed the maximum Subscriber Certificate validity period (currently 398 days; 200 days on or after 2026-03-15; 100 days on or after 2027-03-15; 47 days on or after 2029-03-15 per Section 6.3.2) and SHOULD NOT be later than the <code>notAfter</code> of the Issuing CA Certificate.
<code>tlsCertificate.subject</code>	—	MUST	—	RDNSequence where each <code>RelativeDistinguishedName</code> contains a single <code>AttributeTypeAndValue</code> ; MAY be an empty <code>SEQUENCE</code> . If present, RDNs MUST be encoded in the exact relative order listed in the sub-rows below (<code>countryName</code> → <code>commonName</code>). At most one instance of each attribute is permitted across all RDNs. Attributes MUST NOT contain only metadata such as <code>*,</code> <code>*,*</code> , or <code>*,*</code> (space) characters, or any other indication that the value is absent, incomplete, or not applicable. ASN.1 length limits for <code>DirectoryString</code> are expressed as character limits, not byte limits. Only the attributes listed below are permitted:
<code>L. subject:countryName</code>	2.5.4.6	MAY	—	<code>PrintableString</code> (2 chars). Two-letter ISO 3166-1 alpha-2 country code (or XX if not represented by an official code) for the country associated with the Subject (verified per Section 3.2.2.3).
<code>L. subject:commonName</code>	2.5.4.3	NOT RECOMMENDED	—	<code>UTF8String</code> or <code>PrintableString</code> (max 64 chars). If present, MUST contain exactly one entry derived from <code>subjectAltName</code> : • <code>IPv4 address</code> : encoded as <code>IPv4Address</code> per RFC 3986, Section 3.2.2 . • <code>IPv6 address</code> : encoded in text representation per RFC 5952, Section 4 . • <code>FQDN or Wildcard Domain Name</code> : character-for-character copy of the <code>dNSName</code> entry from <code>subjectAltName</code> (all labels encoded as LDH Labels; P-Labels MUST NOT be converted to Unicode).
<code>L. subject: any other attribute</code>	—	MUST NOT	—	MUST NOT be present.
<code>tlsCertificate.subjectPublicKeyInfo</code>	—	MUST	—	Encoded using one of the following algorithms and exact DER encodings (no other algorithms or key sizes are permitted):
<code>L. RSA Public Key</code>	1.2.848.113549.1.1.1	Conditional	—	Modulus size when encoded MUST be ≥ 2048 bits and evenly divisible by 8; public exponent MUST be an odd integer ≥ 3 (SHOULD be in the range 2 ¹⁶ + 1 to 2 ³⁰ − 1); the CA SHOULD confirm that the modulus is an odd integer, not the power of a prime, and has no factors smaller than 752 (NIST SP 800-89, Section 5.3.3); explicit NULL parameters (0500). Exact AlgorithmIdentifier DER hex encoding: 300906892a864886f78001010500a11c301a06092a064806f780010108100806090686480165830402010500a203020130 (id-RSASSA-PSS 1.2.848.113549.1.1.10 MUST NOT be used).
<code>L. ECDSA Public Key</code>	1.2.848.10045.2.1	Conditional	—	Key MUST represent a valid point on the NIST P-256, P-384, or P-521 elliptic curve, and the CA SHOULD confirm its validity using the ECC Full or Partial Public Key Validation Routine (NIST SP 800-56A Rev. 2, Sections 5.6.2.3.2 / 5.6.2.3.3); <code>namedCurve</code> parameters; uncompressed point format (0x04). Exact AlgorithmIdentifier DER hex encoding: • P-256 (<code>secp256r1</code> , 1.2.848.10045.3.1.7): 301306072a0648cc30020106082a0648cc30030107 • P-384 (<code>secp384r1</code> , 1.3.132.0.34): 301006072a0648cc30020106082a0648cc30030107 • P-521 (<code>secp521r1</code> , 1.3.132.0.35): 301006072a0648cc30020106082a0648cc30030107
<code>tlsCertificate.issuerUniqueID</code>	—	MUST NOT	—	MUST NOT be present.
<code>tlsCertificate.subjectUniqueID</code>	—	MUST NOT	—	MUST NOT be present.
<code>tlsCertificate.extensions</code>	—	MUST	—	X.509v3 extensions SEQUENCE. No extension OID may appear more than once.
<code>authorityInformationAccess (AIA)</code>	1.3.6.1.5.5.7.1.1	SHOULD	N	SEQUENCE of one or more <code>AccessDescription</code> entries. Multiple <code>AccessDescription</code> s with the same <code>accessMethod</code> MAY be present; where they are, each <code>accessLocation</code> MUST be unique and each <code>AccessDescription</code> MUST be ordered in priority for that <code>accessMethod</code> , with the most-preferred <code>accessLocation</code> first. No ordering requirement applies between <code>AccessDescription</code> s with different <code>accessMethod</code> s. Only the access methods listed below are permitted:
<code>L. id-ad-ocsp</code>	1.3.6.1.5.5.7.48.1	MAY	—	<code>uniformResourceIdentifier</code> (<code>IASString</code>) HTTP URL of the Issuing CA's OCSP responder.
<code>L. id-ad-caIssuers</code>	1.3.6.1.5.5.7.48.2	SHOULD	—	<code>uniformResourceIdentifier</code> (<code>IASString</code>) HTTP URL of the Issuing CA certificate.
<code>L. any other accessMethod</code>	—	MUST NOT	—	MUST NOT be present.
<code>authorityKeyIdentifier (AKI)</code>	2.5.29.35	MUST	N	<code>keyIdentifier</code> MUST be present and identical to the <code>subjectKeyIdentifier</code> field of the Issuing CA. <code>authorityCertIssuer</code> and <code>authorityCertSerialNumber</code> MUST NOT be present.
<code>basicConstraints</code>	2.5.29.19	MUST NOT	—	MUST NOT be present.
<code>certificatePolicies</code>	2.5.29.32	MUST	N	SEQUENCE of <code>PolicyInformation</code> entries governed by the rules below:
<code>L. Reserved Policy OID</code>	2.23.148.1.2.1	MUST	—	MUST be present. The extension MUST contain exactly one Reserved Certificate Policy Identifier (2.23.148.1.2.1, Domain Validated). RECOMMENDED as the first <code>PolicyInformation</code> entry ("first_policy_note")
<code>L. anyPolicy</code>	2.5.29.32.0	MUST NOT	—	MUST NOT be present.
<code>L. CA-Defined Policy OIDs</code>	—	MAY	—	Permitted if defined and documented in the CA's CP/CPS.

D. Sunset several profiles



- Update [Section 7.1](#) (“Certificate profile”)
 - Remove all instances of “IV” certificate tables / do not create a consolidated table for IV certificates.
 - Further codify the use of dedicated hierarchies
 - Hierarchies claiming adherence to the TLS BRs must be dedicated exclusively to TLS. Phased implementation where any unexpired non-TLS Sub CA (e.g., S/MIME, Code Signing) chaining to a TLS BR adhering root MUST be revoked.
- Why (perceived ecosystem benefits):
 - [IV] Eliminate complexity incurred by a historically unused profile.
 - [IV] Reduce PII exposure with unclear value proposition.
 - [DEDICATED] Promote simplicity and separation across use cases, align with root stores
 - [DEDICATED] Eliminate [exemptions](#) associated with “constrained” CAs

Intentional deviations and impacts



- Sunset IV Certificates
 - **Zero** Chrome-trusted certificates contain the IV policy OID observed in CT.
 - Censys finds 8 time-valid entries. Only **2** of these have “Browser Trust” - however are clearly test certificates (CN includes “testivg1”) - and these expire next week.

Intentional deviations and impacts [cont]



- Use of name constraints in CA Certificates
 - Related to [sunsetting](#) the concept of “technically constrained”
 - **3** CA certificates trusted in Chrome that contain name constraints.
 - Detailed results: [here](#)

Intentional deviations and impacts [cont]



- Prohibit SCTs and CT Poison extension from appearing in CA Certificates
 - Of the 59 CA certificates found, 0 are trusted in Chrome.
 - Detailed results: [here](#)

Intentional deviations and impacts [cont]



- Prohibit Subscriber and OCSP Responder Certificates from containing the BasicConstraints extension
 - Today, this is a widely common practice, though it's unclear why.
 - [RFC 5280](#) makes the absence of basicConstraints in a version 3 certificate semantically identical to cA=FALSE.
 - Detailed results: [\[1\]](#), [\[2\]](#), and [\[3\]](#)

Intentional deviations and impacts [cont]



- Require Subscriber Certificates contain the keyUsage field
 - Today, keyUsage's presence is SHOULD.
 - In practice, all CA Owners trusted in Chrome include it.
 - Proposed:
 - RSA: digitalSignature (SHOULD), keyEncipherment (MAY)*.
 - EC: digitalSignature (MUST)
 - Notably, this ends 4+ year “[Pending Prohibitions](#)”
 - RSA: [dataEncipherment](#)
 - EC: [keyAgreement](#)
 - Detailed results: [\[1\]](#) and [\[2\]](#)

Intentional deviations and impacts [cont]



- Prohibit Subscriber Certificates from containing the subjectKeyIdentifier extension
 - Per RFC 5280, subjectKeyIdentifier facilitates path construction by matching a child certificate's authorityKeyIdentifier to its issuer's subjectKeyIdentifier.
 - End-entity Subscriber certificates never sign child certificates, so their SKI is never consumed by TLS clients.
 - When included, wastes ~22–35 bytes per leaf certificate transmitted in every TLS handshake.

Discuss

- Proposal
- Alternative solutions
- Next steps
- Blue-sky solutions [time-permitting]

Illustrative Solution #1 - JSON Profile



- Links:
 - **Schema:** <https://gist.github.com/ryancdickson/d8c875c64b96f907bc4433e209003d5a>
 - **Example:** <https://gist.github.com/ryancdickson/b2def106e9adbc6ef3b009b9c201e492>
- How it was built (by a machine):
 - **Profile Schema:**
 - Initially, built for use with the MTC ecosystem.
 - To allow applicability today, transposed all defined fields/extensions permitted by TLS BRs in a JSON schema. Attempted to define “simple” processing constraints.
 - Further improved by considering real-world leafs trusted in Chrome (1000 per CA Owner).
 - **Example Profile:**
 - Normalized and transposed real-world leaf-data into the schema.

Illustrative Solution #1 - JSON Profile [cont]



- How it could be used:
 - Profiles can be bootstrapped from an existing corpus.
 - The JSON profile could be converted into a custom linter (e.g., Python) that verifies presence (or not) of defined fields and extensions, in addition to processing constraints on those objects (e.g., checking certificate validity $\leq$ profile-defined value).
 - Pressure-tested this by creating a corpus-derived profile of real-world misissuance and verified this approach correctly produced a compliant profile (this was expected).

Illustrative Solution #2 - “Differential” JSON Profile



- Links:
 - **Schema:** <https://gist.github.com/ryancdickson/d4b34009214423d917e880d553c4fa6a>
 - **Example:** <https://gist.github.com/ryancdickson/7ef694ee69d0926d8edeb15f70bf7873>
- How it was built (by a machine):
 - **Profile Schema:** Less of a schema, more of a structured inheritance model:
 - **Resolve:** state conditional rules (narrow a “MAY” to a “MUST” or “MUST NOT”)
 - **Constrain:** supply local values (e.g., validity periods, AIA / CRLDP URIs)
 - **Declare:** extensions not included in the TLS BRs
 - **Example Profile:**
 - Normalized and transposed real-world leaf-data into the schema.

Illustrative Solution #2 - “Differential” JSON [cont]



- Opinion: Why Self-Contained (#1) “beats” Differential (#2):
 - Differential profiles (#2) recreate SC-62's core problem - forcing community members and linting tools to join a base BR table with a delta file (which could silently drift whenever the BR version updates).
 - A CP/CPS should be an explicit, standalone contract of the CA's issuance practices rather than a list of exceptions to another document.
 - A self-contained profile (#1) gives CA Owners a single source of truth that can both auto-generate their human-readable CP/CPS Section 7.1 tables and compile directly into their pre-sign & self-audit linters.