RPKI Canonical Cache Representation (CCR)

Job Snijders <job@bsd.nl>
Bart Bakker <bbakker@ripe.net>
Tim Bruijnzeels <tim@ripe.net>
Theo Buehler <tb@openbsd.org>

SIDROPS
IETF 124
Montreal, Canada

Do you often find yourself wondering. . .

- How to reliably compare different transport protocol outcomes?
- How to collate different moments in time?
- How to contrast validation states from different cache implementations?

If any of the above resonated with you, then CCR is for you!

What is Canonical Cache Representation (CCR)?

- CCR is a side quest of the *Erik Synchronisation* project
- A way to represent aspects of an RPKI cache's internal state
- DER-encoded data interchange format defined with ASN.1
- CCR files contain hashes computed over canonicalized data
- Every CCR `ManifestState` is a merkle tree
- Strong embedded file integrity verification
- CCR can be easily be converted to JSON!
  (`rpki-client -j -f ./file.ccr | jq -c ...`)
- CCR data is ***not*** signed data

ASN.1 RpkiCanonicalCacheRepresentation

```
RpkiCanonicalCacheRepresentation ::= SEQUENCE {
  version      [0] INTEGER DEFAULT 0,
  hashAlg          DigestAlgorithmIdentifier,
  producedAt       GeneralizedTime,
  mfts         [1] ManifestState OPTIONAL,
  vrps         [2] ROAPayloadState OPTIONAL,
  vaps         [3] ASPAPayloadState OPTIONAL,
  tas          [4] TrustAnchorState OPTIONAL,
  rks          [5] RouterKeyState OPTIONAL,
  ... }
  ( WITH COMPONENTS { ..., mfts PRESENT } |
    WITH COMPONENTS { ..., vrps PRESENT } |
    WITH COMPONENTS { ..., vaps PRESENT } |
    WITH COMPONENTS { ..., tas PRESENT } |
    WITH COMPONENTS { ..., rks PRESENT } )
```

ASN.1 CCR ManifestState

```
ManifestState ::= SEQUENCE {
  mis                 SEQUENCE OF ManifestInstance,
  mostRecentUpdate    GeneralizedTime,
  hash                Digest }

ManifestInstance ::= SEQUENCE {
  hash                Digest,
  size                INTEGER (1000..MAX),
  aki                 KeyIdentifier,
  manifestNumber      INTEGER (0..MAX),
  thisUpdate          GeneralizedTime,
  locations           SEQUENCE SIZE (1..MAX) OF
                          AccessDescription,
  subordinates        SEQUENCE (SIZE(1..MAX)) OF
                          SubjectKeyIdentifier OPTIONAL }
```

ASN.1 CCR ROAPayloadSet

```
IMPORTS
  ASID, ROAIPAddressFamily
  FROM RPKI-ROA-2023 -- in [RFC9582]
    { iso(1) member-body(2) us(840) rsadsi(113549)
      pkcs(1) pkcs9(9) smime(16) mod(0)
      id-mod-rpkiROA-2023(75) };

ROAPayloadSet ::= SEQUENCE {
  asID           ASID,
  ipAddrBlocks   SEQUENCE (SIZE(1..2)) OF
                     ROAIPAddressFamily }
```

Use cases for CCR

- Feed validated payloads into RTR servers
- Diagnostics & debugging facility
- Easy way to implement record keeping for audit trails
- Monitor precertificates / post validation deltas
- Compare Rsync, RRDP, and Erik transport protocols
- Measure congruence of topologically diverse caches

Take a peek (source refreshed every few minutes)

```
$ curl -Os https://console.rpki-client.org/rpki.ccr.gz

$ gunzip -v rpki.ccr.gz
rpki.ccr.gz: 51.3% -- replaced with rpki.ccr
9966321 bytes in, 20460311 bytes out

$ rpki-client -f rpki.ccr | head
File:                    rpki.ccr
Hash identifier:         SlR8TmFwsYMKjiHQvu7QA0k5zBWTRM00A...
CCR produced at:         Sun 02 Nov 2025 14:49:39 +0000
Manifest state hash:     OENGNkNBNTU4MTFDNEZGRDA1RkI5NjgxR...
Manifest last update:    Sun 02 Nov 2025 14:44:46 +0000
Manifest instances:      hash:AAA2Lx85Gq8Yd06S7CesD9I1K9tW...
                         size:2193
                         aki:3B6FAA2ADB30433D1C1B2F5A84C4E...
                         seqnum:057D
                         thisupdate:1762061378
                         sia:rsync://rpki-repo.registro.br...
```

Next steps

- Review draft-spaghetti-sidrops-rpki-ccr
- Develop more software tooling to interact with CCRs
- Obtain IANA assigned CMS content type OIDs (or squat?)
- WG Adoption $\rightarrow$ Implementation Reports $\rightarrow$ WGLC $\rightarrow$ Profit!