SIDROPS
Internet-Draft
Intended status: Standards Track
Expires: 19 November 2026

J. Snijders
BSD
B. Bakker
T. Bruijnzeels
RIPE NCC
T. Buehler
OpenBSD

18 May 2026

A Profile for Resource Public Key Infrastructure (RPKI) Canonical Cache Representation (CCR)
draft-ietf-sidrops-rpki-ccr-05

Abstract

This document specifies a Canonical Cache Representation (CCR) content type for use with the Resource Public Key Infrastructure (RPKI). CCR is a DER-encoded data interchange format which can be used to represent various aspects of the state of a validated RPKI cache at a particular point in time. The CCR profile is a compact and versatile format well-suited for applications such as audit trails, analytics pipelines, and validated payload dissemination.

Table of Contents

1. Introduction

Resource Public Key Infrastructure (RPKI) operators often wish to analyze Certification Authority (CA) and Relying Party (RP) behavior by inspecting validation outcomes. To this end, Canonical Cache Representation (CCR) was developed to capture and archive RPKI validation states in a standardized data representation.

CCR offers a compact and versatile format well-suited for applications such as audit trails, analytics pipelines, and validated payload dissemination. A validated cache contains all RPKI objects that the RP has verified to be valid according to the rules for validation (see [RFC6487], [RFC6488], [RFC9286]). CCR is a data interchange format using Distinguished Encoding Rules (DER, [X.690]) which can be used to represent various aspects of the state of a validated cache at a particular point in time in a reproducible manner.

This document formally specifies the CCR content type for use with the RPKI and provides test vectors.

1.1. History

The format was initially designed to support comparative analysis of multiple RP instances using a variety of RPKI transport protocols ([RFC5781], [RFC8182], and [I-D.ietf-sidrops-rpki-erik-protocol]).

1.2. Requirements Language

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

2. The Canonical Cache Representation content type

The content of a CCR file is an instance of ContentInfo.

The contentType for a CCR is defined as id-ct-rpkiCanonicalCacheRepresentation, with Object Identifier (OID) 1.2.840.113549.1.9.16.1.54.

The content field contains an instance of RpkiCanonicalCacheRepresentation.

3. The Canonical Cache Representation content

The content of a Canonical Cache Representation is formally defined as follows:

```
RpkiCanonicalCacheRepresentation-2025
  { iso(1) member-body(2) us(840) rsadsi(113549)
    pkcs(1) pkcs9(9) smime(16) mod(0) id-mod-rpkiCCR-2025(TBD) }

DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS
  CONTENT-TYPE, Digest, DigestAlgorithmIdentifier,
    SubjectKeyIdentifier
  FROM CryptographicMessageSyntax-2010 -- in [RFC6268]
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs-9(9) smime(16) modules(0) id-mod-cms-2009(58) }

  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) }

  CAS, PAS
  FROM RPKI-ASPA-2023 -- in [draft-ietf-sidrops-aspa-profile]
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs-9(9) smime(16) modules(0) id-mod-rpki-aspa-2023(TBD) }

  CertificateSerialNumber, SubjectPublicKeyInfo
  FROM PKIX1Explicit-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-explicit-02(51) }

  AccessDescription, KeyIdentifier
  FROM PKIX1Implicit-2009
    { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) id-mod(0)
      id-mod-pkix1-implicit-02(59) }
  ;

ContentInfo ::= SEQUENCE {
  contentType      CONTENT-TYPE.&id({ContentSet}),
  content      [0] EXPLICIT
                   CONTENT-TYPE.&Type({ContentSet}{@contentType}) }

ContentSet CONTENT-TYPE ::= {
```

```
    ct-rpkiCanonicalCacheRepresentation, ... }

  ct-rpkiCanonicalCacheRepresentation CONTENT-TYPE ::=
    { TYPE RpkiCanonicalCacheRepresentation
      IDENTIFIED BY id-ct-rpkiCanonicalCacheRepresentation }

  id-ct-rpkiCanonicalCacheRepresentation OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
      pkcs-9(9) id-smime(16) id-ct(1) ccr(54) }

  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,
    ... }
    -- at least one of mfts, vrps, vaps, tas, or rks MUST be present
    ( WITH COMPONENTS { ..., mfts PRESENT } |
      WITH COMPONENTS { ..., vrps PRESENT } |
      WITH COMPONENTS { ..., vaps PRESENT } |
      WITH COMPONENTS { ..., tas PRESENT } |
      WITH COMPONENTS { ..., rks PRESENT } )

  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 }

  ROAPayloadState ::= SEQUENCE {
    rps               SEQUENCE OF ROAPayloadSet,
    hash              Digest }

  ROAPayloadSet ::= SEQUENCE {
    asID              ASID,
```

```
    ipAddrBlocks      SEQUENCE (SIZE(1..2)) OF ROAIPAddressFamily }

  ASPAPayloadState ::= SEQUENCE {
    aps               SEQUENCE OF ASPAPayloadSet,
    hash              Digest }

  ASPAPayloadSet ::= SEQUENCE {
    customerASID      CAS,
    providers         SEQUENCE (SIZE(1..MAX)) OF PAS }

  TrustAnchorState ::= SEQUENCE {
    skis              SEQUENCE (SIZE(1..MAX)) OF SubjectKeyIdentifier,
    hash              Digest }

  RouterKeyState ::= SEQUENCE {
    rksets            SEQUENCE OF RouterKeySet,
    hash              Digest }

  RouterKeySet ::= SEQUENCE {
    asID              ASID,
    routerKeys        SEQUENCE (SIZE(1..MAX)) OF RouterKey }

  RouterKey ::= SEQUENCE {
    ski               SubjectKeyIdentifier,
    spki              SubjectPublicKeyInfo }

  END
```

3.1. version

The version field contains the format version for the RpkiCanonicalCacheRepresentation structure, in this version of the specification it MUST be 0.

3.2. hashAlg

The hashAlg field specifies the algorithm used to construct the message digests. This profile uses SHA-256 [SHS], therefore the OID MUST be 2.16.840.1.101.3.4.2.1 and the parameters field MUST be absent (Section 2 of [RFC5754]).

3.3. producedAt

The producedAt field contains a GeneralizedTime and indicates the moment in time the CCR was generated. For the purposes of this section, CCR generation begins once the RP’s fetching and validation operations are completed.

3.4. State aspect fields

Each CCR contains one or more fields representing particular aspects of the cache's state. Implementers should note the ellipsis extension marker in the RpkiCanonicalCacheRepresentation ASN.1 notation and anticipate future changes as new signed object types are standardized.

Each state aspect generally consists of a sequence of details extracted from RPKI Objects of a specific type, along with a digest computed by hashing the aforementioned DER-encoded sequence, and optionally including some metadata.

3.4.1. ManifestState

An instance of ManifestState represents the set of valid, current Manifests ([RFC9286]) in the cache. It contains three fields: mis, mostRecentUpdate, and hash.

3.4.1.1. ManifestInstance

The mis field contains a SEQUENCE of ManifestInstance. There is one ManifestInstance for each current manifest. A manifest is nominally current until the time specified in nextUpdate, or until a manifest is issued with a greater manifestNumber (see Section 4.2.1 of [RFC9286]), or until a new manifest is issued with a new filename per the process described in section 2 of [RFC9981].

A ManifestInstance is a structure consisting of the following fields:

hash the hash of the represented DER-encoded manifest object

size the size of the represented DER-encoded manifest object

aki the manifest issuer's key identifier

manifestNumber the manifest number contained within the manifest's eContent field

thisUpdate the thisUpdate contained within the manifest's eContent field

locations a sequence of AccessDescription instances from the manifest's End-Entity certificate's Subject Information Access extension

subordinates a optional non-empty SEQUENCE of SubjectKeyIdentifier

The subordinates field represents the keypairs associated with the set of non-revoked, non-expired, validly signed, certification authority (CA) resource certificates subordinate to the manifest issuer. Each SubjectKeyIdentifier is the 160-bit SHA-1 hash of the value of the DER-encoded ASN.1 bit string of the resource certificate's Subject Public Key, as described in Section 4.8.2 of [RFC6487]. The sequence elements of the subordinates field MUST be sorted in ascending order by interpreting each SubjectKeyIdentifier value as an unsigned 160-bit integer and MUST be unique with respect to each other.

The sequence elements in the mis field MUST be sorted in ascending order by hash value contained in each instance of ManifestInstance and MUST be unique with respect to the other instances of ManifestInstance.

3.4.1.2. mostRecentUpdate

The mostRecentUpdate is a metadata field which contains the most recent thisUpdate amongst all current manifests represented by the ManifestInstance structures. If the mis field contains an empty sequence, the mostRecentUpdate MUST be set to the POSIX Epoch ("19700101000000Z").

The above and the requirements in Section 6.3 of [RFC9286] imply that mostRecentUpdate MUST precede or be equal to producedAt (Section 3.3).

3.4.1.3. hash

The hash field contains a message digest computed using the mis value (encoded in DER format) as input message.

3.4.2. ROAPayloadState

An instance of ROAPayloadState contains a field named rps which represents the current set of Validated ROA Payloads (Section 2 of [RFC6811]) encoded as a SEQUENCE of ROAPayloadSet instances ordered by ascending asID.

The ROAPayloadSet structure is modeled after the RouteOriginAttestation (Section 4 of [RFC9582]). The asID value in each instance of ROAPayloadSet MUST be unique with respect to other instances of ROAPayloadSet. The contents of the ipAddrBlocks field MUST appear in canonical form and ordered as defined in Section 4.3.3 of [RFC9582].

The hash field contains a message digest computed using the rps value (encoded in DER format) as input message.

3.4.3. ASPAPayloadState

An instance of ASPAPayloadState contains an aps field which represents the current set of deduplicated and merged ASPA payloads ([I-D.ietf-sidrops-aspa-profile]) value encoded as a SEQUENCE of ASPAPayloadSet instances ordered by ascending customerASID. The customerASID value in each instance of ASPAPayloadSet MUST be unique with respect to other instances of ASPAPayloadSet.

The ASPAPayloadSet structure is modeled after the ProviderASSet (Section 3.3 of [I-D.ietf-sidrops-aspa-profile]). The elements of providers MUST be ordered in ascending numerical order and MUST be unique (with respect to the other elements of providers). A PAS value of 0 can only be encoded in the providers field as a single item list, i.e., an element for AS 0 MUST NOT appear alongside any other elements.

The hash field contains a message digest computed using the aps value (encoded in DER format) as input message.

3.4.4. TrustAnchorState

An instance of TrustAnchorState represents the set of valid Trust Anchor (TA) Certification Authority (CA) resource certificates used by the relying party when producing the CCR.

Each SubjectKeyIdentifier is the 160-bit SHA-1 hash of the value of the DER-encoded ASN.1 bit string of the TA’s Subject Public Key, as described in Section 4.8.2 of [RFC6487]. The skis field contains a sequence of Subject Key Identifiers (SKI) sorted in ascending order by interpreting the SKI value as an unsigned 160-bit integer.

The hash field contains a message digest computed using the skis value (encoded in DER format) as input message.

3.4.5. RouterKeyState

An instance of RouterKeyState contains an rksets field which represents the current set of valid BGPsec Router Keys [RFC8205] encoded as a SEQUENCE of RouterKeySet instances. The asID value in each instance of RouterKeySet MUST be unique with respect to other instances of RouterKeySet. Instances of RouterKeySet are sorted by ascending value of asID. Instances of RouterKey are sorted by ascending value of ski by interpreting the SKI value as an unsigned 160-bit integer.

The hash field contains a message digest computed using the rks value (encoded in DER format) as input message.

4. Use Cases

This section describes a number of applications for the CCR format across different contexts.

4.1. Constructing Consistent Views on Distributed Data

This section describes a use case for CCRs in the context of distributed systems.

Assuming CAs issue Manifests in accordance with Section 5 of [RFC9286], a ManifestInstance can be considered a state-based Conflict-free Replicated Data Type ([CRDT]), meaning that ManifestInstance sets contain sufficient information to form a monotonic semilattice.

The implication is that ManifestState instances from multiple CCRs produced by multiple different RPs at different times can safely be merged in order to construct an internally consistent view of the RPKI distributed database.

The reconciled merge result can be useful, for example, as a backend for Erik Synchronization relays ([I-D.ietf-sidrops-rpki-erik-protocol]) which execute separate validation processes for different Trust Anchors and varying maximum certificate chain depths.

4.2. Data Collection

Operators have an interest in determining how the global RPKI is viewed from the perspectives of several different locations around the Internet. As CCR allows for point-in-time capture and later reconstruction and analysis, it found use in multi-perspective collector methods such as described RPKISPOOL [I-D.snijders-rpkispool-format].

An example of a large-scale CCR-based RPKI data archival project is [RPKIViews].

5. Operational Considerations

This section covers operational considerations.

5.1. CCR file integrity

The integrity of a CCR file can be checked by confirming whether the hash value embedded inside each state aspect matches the computed hash value of the respective state aspect payload structure. Readers MUST verify the integrity of CCR files and stop further processing on failure.

5.2. Timing analysis

The producedAt timestamp is not necessarily the current time used by the RP for the purposes of validating the RPKI content. In practice, most RPs interleave fetching and validation operations, with validation occurring with respect to whatever the time happens to be at that point (i.e., wall clock time). This means that it is possible for a CCR to include information that would have been excluded if validated at the time indicated by the producedAt timestamp.

If the CCR is produced right after all relevant repository content was received and validated by an RP, then comparing the ManifestState mostRecentUpdate timestamp (Section 3.4.1.2) value with the CCR producedAt timestamp (Section 3.3) might help offer insight into the timing and propagation delays of the RPKI ecosystem.

5.3. Storage efficiency

CCRs compress very well due to its data layout characteristics: the content contains repetitive sequences, does not contain high entropy data such as public keys, and is consistently ordered. Readers and writers of CCR data are RECOMMENDED to support data compression using Gzip ([RFC1952]) in context of durable storage.

6. Security Considerations

The CCR format utilizes a structure that can store information about the state of a given RPKI cache at a particular moment in time. The fields defined in this specification are of a descriptive nature and provide information that is useful to facilitate the analysis of RPKI data. As such, these fields do not in themselves create additional security risks, since the fields are not used to induce any particular behavior by the recipient application.

Readers MUST check contextual bounds on all fields appropriately and stop further processing on failure. E.g., the maxLength element in a ROAIPAddress cannot contain an integer smaller than the length of the accompanying prefix, the manifestNumber field is cannot be longer than 20 octets, etc.

The CCR format contains no executable code, and it does not define any extensible areas that could be used to store such code.

CCRs are not signed objects. RPKI information is normally public and does not call for confidentiality protection. Ascertaining the provenance (and thus authenticity) of any given CCR is out-of-scope for this document.

7. IANA Considerations

7.1. SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)

IANA has allocated the following in the "SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)" registry:

Decimal	Description	References
54	id-ct-rpkiCanonicalCacheRepresentation	draft-ietf-sidrops-rpki-ccr

Table 1

7.2. RPKI Repository Name Schemes

IANA is requested to add the Canonical Cache Representation file extension to the "RPKI Repository Name Schemes" registry [RFC6481] as follows:

Filename Extension	RPKI Object	Reference
.ccr	Canonical Cache Representation	draft-ietf-sidrops-rpki-ccr

Table 2

7.3. SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)

IANA is requested to allocate the following in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry:

Decimal	Description	References
TBD	id-mod-rpkiCCR-2025	draft-ietf-sidrops-rpki-ccr

Table 3

7.4. Media Types

IANA is requested to register the media types "application/rpki-ccr" and "application/rpki-ccr+gzip" in the "Media Types" registry as follows:

7.4.1. Canonical Cache Representation Media Type

Type name: application
Subtype name: rpki-ccr
Required parameters: N/A
Optional parameters: N/A
Encoding considerations: binary
Security considerations: This media type contains no active content.
Interoperability considerations: N/A
Published specification: draft-ietf-sidrops-rpki-ccr
Applications that use this media type: RPKI operators
Fragment identifier considerations: N/A
Additional information:
Content: This media type is a RPKI Canonical Cache Representation object, as defined in draft-ietf-sidrops-rpki-ccr.
Magic number(s): N/A
File extension(s): .ccr
Macintosh file type code(s): N/A
Person & email address to contact for further information: Job Snijders (job@bsd.nl)
Intended usage: COMMON
Restrictions on usage: N/A
Author: Job Snijders (job@bsd.nl)
Change controller: IETF

Type name: application
Subtype name: rpki-ccr+gzip
Content: This media type is a Gzip compressed RPKI Canonical Cache Representation object, as defined in draft-ietf-sidrops-rpki-ccr.
Magic number(s): N/A
File extension(s): .ccr.gz
References: RFC1952, RFC6713

Encoding considerations: gzip is a binary encoding

8. References

8.1. Normative References

[I-D.ietf-sidrops-aspa-profile]
Snijders, J., Azimov, A., Uskov, E., Bush, R., Housley, R., and B. Maddison, "A Profile for Autonomous System Provider Authorization", Work in Progress, Internet-Draft, draft-ietf-sidrops-aspa-profile-26, 19 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-profile-26>.

[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>.

[RFC5754] Turner, S., "Using SHA2 Algorithms with Cryptographic Message Syntax", RFC 5754, DOI 10.17487/RFC5754, January 2010, <https://www.rfc-editor.org/info/rfc5754>.

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <https://www.rfc-editor.org/info/rfc6481>.

[RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <https://www.rfc-editor.org/info/rfc6487>.

[RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, DOI 10.17487/RFC6488, February 2012, <https://www.rfc-editor.org/info/rfc6488>.

[RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, DOI 10.17487/RFC6811, January 2013, <https://www.rfc-editor.org/info/rfc6811>.

[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>.

[RFC9286] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 9286, DOI 10.17487/RFC9286, June 2022, <https://www.rfc-editor.org/info/rfc9286>.

[RFC9582] Snijders, J., Maddison, B., Lepinski, M., Kong, D., and S. Kent, "A Profile for Route Origin Authorizations (ROAs)", RFC 9582, DOI 10.17487/RFC9582, May 2024, <https://www.rfc-editor.org/info/rfc9582>.

[RFC9981] Harrison, T., Michaelson, G., and J. Snijders, "Resource Public Key Infrastructure (RPKI) Manifest Number Handling", May 2026, <https://www.rfc-editor.org/info/rfc9981>.

[SHS] National Institute of Standards and Technology, "Secure Hash Standard", March 2012, <https://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[X.690] ITU-T, "Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2021, February 2021, <https://www.itu.int/rec/T-REC-X.690-202102-I/en>.

8.2. Informative References

[CRDT] Shapiro, M., Pregui巽a, N., Baquero, C., and M. Zawirski, "Conflict-free Replicated Data Types", INRIA RR-7687, July 2011, <https://inria.hal.science/inria-00609399>.

[I-D.ietf-sidrops-rpki-erik-protocol] Snijders, J., Bruijnzeels, T., Harrison, T., and W. Ohgai, "The Erik Synchronization Protocol for use with the Resource Public Key Infrastructure (RPKI)", Work in Progress, Internet-Draft, draft-ietf-sidrops-rpki-erik-protocol-04, 17 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-rpki-erik-protocol-04>.

[I-D.snijders-rpkispool-format]
Snijders, J. and F. Vompe, "The RPKISPOOL Format for Materializing Resource Public Key Infrastructure (RPKI) Data", Work in Progress, Internet-Draft, draft-snijders-rpkispool-format-00, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-snijders-rpkispool-format-00>.

[RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, DOI 10.17487/RFC1952, May 1996, <https://www.rfc-editor.org/info/rfc1952>.

[RFC5781] Weiler, S., Ward, D., and R. Housley, "The rsync URI Scheme", RFC 5781, DOI 10.17487/RFC5781, February 2010, <https://www.rfc-editor.org/info/rfc5781>.

[RFC8182] Bruijnzeels, T., Muravskiy, O., Weber, B., and R. Austein, "The RPKI Repository Delta Protocol (RRDP)", RFC 8182, DOI 10.17487/RFC8182, July 2017, <https://www.rfc-editor.org/info/rfc8182>.

[RFC8205] Lepinski, M., Ed. and K. Sriram, Ed., "BGPsec Protocol Specification", RFC 8205, DOI 10.17487/RFC8205, September 2017, <https://www.rfc-editor.org/info/rfc8205>.

[rpki-client]
Jeker, C., Dzonsons, K., Buehler, T., and J. Snijders, "rpki-client", December 2025, <https://www.rpki-client.org/>.

[rpki-commons]
NCC, R., "rpki-commons", April 2026, <https://github.com/RIPE-NCC/rpki-commons>.

[rpkitouch]
Snijders, J., "rpki-client", December 2025, <https://www.github.com/job/rpkitouch>.

[RPKIViews]
Snijders, J., "The RPKIViews Project", April 2026, <https://www.rpkiviews.org/>.

Appendix A. Acknowledgements

The authors wish to thank Russ Housley, Luuk Hendriks, Fedor Vompe, and Tom Harrison for their generous feedback on this specification.

Appendix B. Example CCR

The below is a Base64-encoded example CCR object. For a more elaborate example based on the global RPKI, see the URL in Appendix C.

MIIF9AYLKoZIhvcNAQkQATagggXjMIIF3zALBglghkgBZQMEAgEYDzIwMjYwNTE1MDAwM
DEwWqGCAtUwggLRMIICmjCBmAQgKF60zgHHRNmQSUXcsAcAPB2cB7kvToWUF60GADJuG5
ECAgPpBBSi3wQv6LAAYxHolIUawRQRMHtgQwICEyEYDzIwMjYwNTE1MDAwMDA5WjBFMEM
GCCsGAQUFBzALhjdyc3luYzovL2V4YW1wbGUubmV0L2NhNC9Ra3NiUVpNQzdZV3NOclJF
dDRsNGRXQVExc0UubWZ0MIGYBCA8fzi045g3wS16tiKY4MxriwOP0eQx7JM3IKzL/1D/j
wICB/gEFPrL0CykfjvZZm/L2COzfe3QvO4AAgICAxgPMjAyNjA1MTUwMDAwMDdaMEUwQw
YIKwYBBQUHMAuGN3JzeW5jOi8vZXhhbXBsZS5uZXQvY2EyL3owbnpWUzdTT0JfOXk2dGF
wSGs3LVl1S2ttOC5tZnQwgZgEIL3nuZvothSocx8JXZLAtiF9FpVXBx1btwfKgDJ5Pv16
AgIPmwQU5zFepRXXwgU4aBJJ0+MNZ3cWJYUCAgUIGA8yMDI2MDUxNTAwMDAwOFowRTBDB
ggrBgEFBQcwC4Y3cnN5bmM6Ly9leGFtcGxlLm5ldC9jYTMvc2JoRnp6NHdUcXNGbzJOVl
JNOG1XZnNQQktRLm1mdDCBxgQg48JkKNPGfzSWjkALB4rFbaktXGSFaAV5qj0gj7zCCFY
CAgbBBBQl+Mz878BG2NzQD8DkROCqe3kPlgICAQEYDzIwMjYwNTE1MDAwMDA2WjBFMEMG
CCsGAQUFBzALhjdyc3luYzovL2V4YW1wbGUubmV0L2NhMS9PYVZVT0lEU2FMelViZWl6N
lZQb2dYeHNLNW8ubWZ0MCwEFKLfBC/osABjEeiUhRrBFBEwe2BDBBTnMV6lFdfCBThoEk
nT4w1ndxYlhRgPMjAyNjA1MTUwMDAwMDlaBCBjjUCOSmIWv8DNHb9zxwi1k6YgLC4hpk4
aph0pqidsEqKBoTCBnjB6MBUCAQAwEDAOBAIAATAIMAYDBADAAAIwLQIDAQAAMCYwEQQC
AAEwCzAJAwQAxjNkAgEcMBEEAgACMAswCQMHACABDQgAADAYAgMBAA4wETAPBAIAAjAJM
AcDBQA//wAAMBgCAwEADzARMA8EAgACMAkwBwMFAD//AAAEIJgOVAZ7JE7OekW9qMlKUN
jkGdzgod7Fcobph5AfXVkCo1MwUTAtMAwCAwD7/zAFAgMA+/AwEQIDAQAAMAoCAwEABAI
DAQAIMAoCAwEADjADAgEABCAnN98QySyKCzUlPnxJJT5iGrRQCLLbvCDdt4esCyUUU6RS
MFAwLAQUJfjM/O/ARtjc0A/A5ETgqnt5D5YEFPrL0CykfjvZZm/L2COzfe3QvO4ABCAO5
kLEyVH4bH17eMAESlf9gYYe1a99AfW+q44/jdcDEaWCAZcwggGTMIIBbTCB7gIDAP5jMI
HmMHEEFIjF3ilaMnbWnpu3RpvUbvly3jKsMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgA
E64mxtNmdKd1bxIjgWrGJutr11LDeA56L8cc1NLL/WW9RZ+rbi+G4rFSvfrEjxzRPt6tc
NWpgEINq7tOR7J5dAjBxBBS+FudOEPS98/jCYYsCSpRX37+J+jBZMBMGByqGSM49AgEGC
CqGSM49AwEHA0IABCo6kzaMUiyt1JyzDY9gHTf+bJOUiE52FiuXvyXmypSTxRcZgazl1k
Moo0UARRbrOxrSyyGQWIKeCv7vKNvw+IowegIDAQAPMHMwcQQURgK2IbAXaB5h7h9KXvw
dAsO0bywwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAThe3r51EOGOYfRBWZeVQ+d0l5f
LOUxyxyjpaSuMF/o2hfqBhqERKAKbrvGQErhnGG8JlEVYvGofxyBP8+C+X3jBCCfSt7Zy
MVIWZ18hjoqeDkmVGKSbWfe4VJZrVgJs5v/FA==

It decodes as follows:

=============== NOTE: '\' line wrapping per RFC 8792 ================

```
File:                   example.ccr
Hash identifier:        8mdCrklrbLHKzDVr5fFbsGK4fyJsArmsGeLShZhRyio=
CCR produced at:        Fri 15 May 2026 00:00:10 +0000
Manifest state hash:    Y41AjkpiFr/AzR2/c8cItZOmICwuIaZOGqYdKaonbBI=
Manifest last update:   Fri 15 May 2026 00:00:09 +0000
Manifest instances:
                        hash:48JkKNPGfzSWjkALB4rFbaktXGSFaAV5qj0gj7z\
```

```
   CCFY= size:1729 aki:25F8CCFCEFC046D8DCD00FC0E444E0AA7B790F96 seqnum:\
   0101 thisupdate:1778803206 sia:rsync://example.net/ca1/OaVUOIDSaLzUb\
   eiz6VPogXxsK5o.mft subordinates:A2DF042FE8B0006311E894851AC11411307B\
   6043,E7315EA515D7C20538681249D3E30D6777162585
                           hash:KF60zgHHRNmQSUXcsAcAPB2cB7kvToWUF60GADJ\
   uG5E= size:1001 aki:A2DF042FE8B0006311E894851AC11411307B6043 seqnum:\
   1321 thisupdate:1778803209 sia:rsync://example.net/ca4/QksbQZMC7YWsN\
   rREt4l4dWAQ1sE.mft
                           hash:vee5m+i2FKhzHwldksC2IX0WlVcHHVu3B8qAMnk\
   +/Xo= size:3995 aki:E7315EA515D7C20538681249D3E30D6777162585 seqnum:\
   0508 thisupdate:1778803208 sia:rsync://example.net/ca3/sbhFzz4wTqsFo\
   2NVRM8mWfsPBKQ.mft
                           hash:PH84tOOYN8EterYimODMa4sDj9HkMeyTNyCsy/9\
   Q/48= size:2040 aki:FACBD02CA47E3BD9666FCBD823B37DEDD0BCEE00 seqnum:\
   0203 thisupdate:1778803207 sia:rsync://example.net/ca2/z0nzVS7SOB_9y\
   6tapHk7-YuKkm8.mft
   ROA payload state hash: mA5UBnskTs56Rb2oyUpQ2OQZ3OCh3sVyhumHkB9dWQI=
   ROA payload entries:
                           192.0.2.0/24 AS 0
                           198.51.100.0/24-28 AS 65536
                           2001:d08::/48 AS 65536
                           3fff::/32 AS 65550
                           3fff::/32 AS 65551
   ASPA payload state hash:JzffEMksigs1JT58SSU+Yhq0UAiy27wg3beHrAslFFM=
   ASPA payload entries:
                           customer: 64511 providers: 64496
                           customer: 65536 providers: 65540, 65544
                           customer: 65550 providers: 0
   Trust anchor state hash:DuZCxMlR+Gx9e3jABEpX/YGGHtWvfQH1vquOP43XAxE=
   Trust anchor keyids:    25F8CCFCEFC046D8DCD00FC0E444E0AA7B790F96, FA\
   CBD02CA47E3BD9666FCBD823B37DEDD0BCEE00
   Router key state hash:  n0re2cjFSFmdfIY6Kng5JlRikm1n3uFSWa1YCbOb/xQ=
   Router keys:
                           asid:65123 ski:88C5DE295A3276D69E9BB7469BD46\
   EF972DE32AC pubkey:MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE64mxtNmdKd1bx\
   IjgWrGJutr11LDeA56L8cc1NLL/WW9RZ+rbi+G4rFSvfrEjxzRPt6tcNWpgEINq7tOR7\
   J5dAg==
                           asid:65123 ski:BE16E74E10F4BDF3F8C2618B024A9\
   457DFBF89FA pubkey:MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEKjqTNoxSLK3Un\
   LMNj2AdN/5sk5SITnYWK5e/JebKlJPFFxmBrOXWQyijRQBFFus7GtLLIZBYgp4K/u8o2\
   /D4ig==
                           asid:65551 ski:4602B621B017681E61EE1F4A5EFC1\
   D02C3B46F2C pubkey:MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE4Xt6+dRDhjmH0\
   QVmXlUPndJeXyzlMcsco6WkrjBf6NoX6gYahESgCm67xkBK4ZxhvCZRFWLxqH8cgT/Pg\
   vl94w==
   Validation:             N/A
```

Appendix C. Implementation status

This section is to be removed before publishing as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

* Example .ccr files were created by Job Snijders. A current example CCR (regenerated every few minutes) is available here: https://console.rpki-client.org/rpki.ccr

* A CCR serializer and deserializer implementation based on [rpki-client] was provided by Job Snijders and Theo Buehler.

* Another CCR serializer, deserializer, and CRDT effector implementation based on [rpkitouch] was provided by Job Snijders.

* A CCR encoding and decoding implementation in Java library [rpki-commons] was provided by RIPE NCC.

Authors' Addresses

Job Snijders
BSD Software Development
Amsterdam
Netherlands
Email: job@bsd.nl
URI: https://www.bsd.nl

Bart Bakker
RIPE NCC
Netherlands
Email: bbakker@ripe.net

Tim Bruijnzeels
RIPE NCC
Netherlands
Email: tbruijnzeels@ripe.net

Theo Buehler
OpenBSD
Switzerland
Email: tb@openbsd.org