Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 April 2026

T. Harrison
APNIC
T. Bruijnzeels
RIPE NCC
C. Martinez-Cagnazzo
LACNIC
M. Kosters
ARIN
Y. Chadee
AFRINIC

20 October 2025

RPKI Trust Anchor Constraints
draft-nro-sidrops-ta-constraints-00

Abstract

Resource Public Key Infrastructure (RPKI) Relying Parties (RPs) are commonly configured with five Trust Anchors (TAs), one for each of the Regional Internet Registries (RIRs). Each TA operator is able to make arbitrary RPKI statements about resources independently of the other TA operators: for example, one TA could issue a Route Origin Authorization (ROA) for resources that have actually been assigned to another TA.

This document specifies a protocol that allows a set of TAs to make signed statements that assert their consensus as to the resources for which each TA is authoritative.

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 23 April 2026.

Table of Contents

1. Requirements notation

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. Introduction

2.1. Problem Statement

In the context of the Resource Public Key Infrastructure (RPKI) [RFC6480], validation is performed by Relying Party (RP) software. RPs are configured with one or more Trust Anchor Locators (TALs) [RFC8630], each of which contains the public key and URI(s) for an RPKI Trust Anchor (TA) certificate. An RP uses its configured TALs to retrieve and validate the RPKI content published by the associated TAs. In a conventional production configuration, an RP is configured with five TALs, one for each of the Regional Internet Registries (RIRs).

TALs operate as a layer of indirection, permitting a TA to reissue its certificate and publish it at the URL(s) specified in the TAL without requiring RPs to update their configuration. However, since TALs do not themselves restrict the Internet Number Resources (INRs)

that may be claimed by a given TA, it is possible for a TA to reissue its certificate with an arbitrary set of INRs, and have RPs trust the TA with respect to those resources without any intervention on the part of the RP's operator. This aspect of how TALs function leads to the possibility of a TA claiming resources for which it is not considered authoritative.

2.2. Goal of this Specification

The main goal of this specification is to prevent TAs from claiming resources for which they are not authoritative. This is achieved by having a set of TAs issue various signed objects that attest to their shared understanding of the resources for which each TA is authoritative.

2.3. Glossary

This section describes the terminology and abbreviations used in this document.

* Internet Number Resources (INRs)

IPv4 addresses, IPv6 addresses, and Autonomous System Numbers (ASNs).

* Resource Public Key Infrastructure (RPKI)

The RPKI is the core infrastructure that is used to associate public keys with specific INRs, allowing the holders of the corresponding private keys to make attestations about those INRs, such as Route Origin Authorizations (ROAs) [RFC9582].

* Trust Anchor (TA)

An apex of trust in the RPKI hierarchy. Trust Anchors (TAs) issue a self-signed RPKI CA Certificate [RFC6487] which enumerates the INRs for which the TA considers itself authoritative.

* TA Constraints

For a set of TAs participating in a consensus, the details on the INRs for which each TA is considered authoritative by that consensus.

* Participating TAs / Participants

TAs that participate together, by way of the mechanisms described in this document, in agreeing on constraints that apply to each participant.

* Constraint Validator

An entity that observes and validates constraint messages signed and published by participants, in order to determine the INRs for which the set of participants considers each TA authoritative.

2.4. Requirements

This section describes the requirements, divided into topical subsections, that were taken into account in the design of this specification.

2.4.1. Signing Outside of RPKI Repository

Each current RIR TA issues a TA certificate that enumerates the complete set of INRs (i.e. 0/0 for IPv4, ::/0 for IPv6, and 1-4294967295 for ASNs). This is because the alternative approach of enumerating the resources for which the TA considers itself authoritative would require that the TA be available for online signing, in order to handle inter-RIR transfers in a timely manner, and offline signing for the TA is preferred by the RIRs for various reasons. The constraint validation process must operate in such a way that TAs can continue to enumerate the complete set of INRs, notwithstanding that constraint validation will apply additional restrictions with respect to the set of INRs for which a given TA may make statements.

Participating TAs must be able to use an offline signing process for the RPKI TA certificate, and delegate the responsibility of signing constraint-related statements to another key that can be used online.

The impact on the RPKI in terms of signed objects should be minimised, in order to avoid overhead or issues with RPKI RP software that is unable or unwilling to use TA constraint objects.

2.4.2. Initiating Participants and Resource Distribution

The TA constraints must be agreed among a set of participating TAs.

The initial set of participants must be agreed on by all participants.

TAs must agree on initial INR constraints that apply to each participant. These INRs cannot overlap.

2.4.3. Robustness

It must not be possible for action taken by a single participant in a consensus to cause an RP to skip constraint validation for the consensus.

2.4.4. Transfers between Participants

The TA that holds an INR in their constraints can transfer it to another TA. For the transfer to take effect, the receiving TA needs to accept the transfer, and the originating TA needs to confirm the transfer. Prior to confirmation, the originating TA must have the option to cancel the transfer.

An INR that is transferred from one TA to another is considered no longer held by the former, and cannot be transferred again by that TA.

An INR that was transferred to a participating TA can be transferred again by that TA to any other participant, including the TA from which the INRs were originally received (local policies permitting).

If an INR transfer was completed in error, then the transfer cannot be reversed. However, the recipient can initiate another transfer to return the INR(s) in question.

2.4.5. Movement of INRs into or out of the Consensus

Any participant can declare itself authoritative for resources that are not currently part of the consensus (i.e. for which another TA is not already considered authoritative).

Any participant can declare that it is no longer authoritative for any INRs for which the consensus currently considers it authoritative.

As with transfers, such declarations cannot be reversed directly, but the same effect can be achieved by making a compensating declaration.

2.4.6. Adding and Removing Participants

New TAs can be included in the set, if all participants agree.

If a TA is no longer trusted by other participants, the remaining participants can remove them from the set, essentially forming a new set that no longer includes that participant.

3. Conventions Used in This Document

Indentation and whitespace in examples are provided only to illustrate element relationships, and are not a REQUIRED feature of this protocol.

"..." in examples is used as shorthand for elements defined outside of this document.

4. Resource Distribution Objects

Resource Distribution Objects comprise Resource Distribution State (RDS) and Resource Distribution Event (RDE) objects. The use of these objects in the wider context of signing and validating constraints is described in the "Protocol Walkthrough" section later in this document.

RDOs have the following formal definition:

```
DEFINITIONS EXPLICIT TAGS ::=
BEGIN

IMPORTS
 CONTENT-TYPE
 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) }

 IPAddressOrRange, ASIdOrRange
 FROM IPAddrAndASCertExtn -- in [RFC3779]
  { iso(1) identified-organization(3) dod(6) internet(1)
   security(5) mechanisms(5) pkix(7) mod(0)
   id-mod-ip-addr-and-as-ident(30) } ;

ct-resourceDistributionState CONTENT-TYPE ::=
 { TYPE ResourceDistributionState
  IDENTIFIED BY id-ct-resourceDistributionState }

id-ct-resourceDistributionState OBJECT IDENTIFIER ::=
 { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
  pkcs-9(9) id-smime(16) id-ct(1) X1 }

ct-transferInitiation CONTENT-TYPE ::=
 { TYPE TransferInitiation
  IDENTIFIED BY id-ct-transferInitiation }

id-ct-transferInitiation OBJECT IDENTIFIER ::=
 { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
```

```
     pkcs-9(9) id-smime(16) id-ct(1) X2 }

   ct-transferAcceptance CONTENT-TYPE ::=
    { TYPE TransferAcceptance
     IDENTIFIED BY id-ct-transferAcceptance }

   id-ct-transferAcceptance OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs-9(9) id-smime(16) id-ct(1) X3 }

   ct-transferFinalisation CONTENT-TYPE ::=
    { TYPE TransferFinalisation
     IDENTIFIED BY id-ct-transferFinalisation }

   id-ct-transferFinalisation OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs-9(9) id-smime(16) id-ct(1) X4 }

   ct-transferCancellation CONTENT-TYPE ::=
    { TYPE TransferCancellation
     IDENTIFIED BY id-ct-transferCancellation }

   id-ct-transferCancellation OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs-9(9) id-smime(16) id-ct(1) X5 }

   ct-resourceInclusion CONTENT-TYPE ::=
    { TYPE ResourceInclusion
     IDENTIFIED BY id-ct-resourceInclusion }

   id-ct-resourceInclusion OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs-9(9) id-smime(16) id-ct(1) X6 }

   ct-resourceExclusion CONTENT-TYPE ::=
    { TYPE ResourceExclusion
     IDENTIFIED BY id-ct-resourceExclusion }

   id-ct-resourceExclusion OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs-9(9) id-smime(16) id-ct(1) X7 }

   URI    ::= IA5String
   TaName   ::= IA5String
   TransferId ::= IA5String
   ResourceInclusionId ::= IA5String
   ResourceExclusionId ::= IA5String
```

```
   IPAddressFamily   ::= SEQUENCE { -- AFI &amp;amp; opt SAFI --
    addressFamily    OCTET STRING , -- (SIZE (2..3)), --
    addresses      SEQUENCE OF IPAddressOrRange }

   IPAddrBlocks    ::= SEQUENCE OF IPAddressFamily

   Delegation ::= SEQUENCE {
    taName        TaName,
    ips          IPAddrBlocks,
    asns         SEQUENCE OF ASIdOrRange }

   ResourceDistributionState ::= SEQUENCE {
    version        INTEGER,
    date         GeneralizedTime,
    previousRDS      URI OPTIONAL,
    urlPrefix       URI,
    rdoIndex       INTEGER OPTIONAL,
    delegations      SEQUENCE OF Delegation }

   TransferInitiation ::= SEQUENCE {
    id          TransferId,
    date         GeneralizedTime,
    recipientTaName    TaName,
    ips          IPAddrBlocks,
    asns         SEQUENCE OF ASIdOrRange }

   TransferAcceptance ::= SEQUENCE {
    transferInitiationId TransferId,
    date         GeneralizedTime,
    sourceTaName     TaName,
    ips          IPAddrBlocks,
    asns         SEQUENCE OF ASIdOrRange }

   TransferFinalisation ::= SEQUENCE {
    transferInitiationId TransferId,
    date         GeneralizedTime }

   TransferCancellation ::= SEQUENCE {
    transferInitiationId TransferId,
    date         GeneralizedTime }

   ResourceInclusion ::= SEQUENCE {
    id           ResourceInclusionId,
    date         GeneralizedTime,
    ips          IPAddrBlocks,
    asns         SEQUENCE OF ASIdOrRange }

   ResourceExclusion ::= SEQUENCE {
```

```
    id          ResourceExclusionId,
    date        GeneralizedTime,
    ips         IPAddrBlocks,
    asns        SEQUENCE OF ASIdOrRange }

END
```

These objects are not modelled as RPKI CMS objects, because in many cases they are not about resources for which the signing TA is actually authoritative. The RDS object itself is an example of this.

5. Resource Distribution Consensus (RDC) Objects

The use of Resource Distribution Consensus (RDC) objects in the wider context of signing and validating constraints is described in the "Protocol Walkthrough" section later in this document.

RDC objects have the following formal definition:

```
   DEFINITIONS EXPLICIT TAGS ::=
   BEGIN

   IMPORTS
    CONTENT-TYPE
    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) }

    IPAddressOrRange, ASIdOrRange
    FROM IPAddrAndASCertExtn -- in [RFC3779]
     { iso(1) identified-organization(3) dod(6) internet(1)
      security(5) mechanisms(5) pkix(7) mod(0)
      id-mod-ip-addr-and-as-ident(30) }

     SubjectPublicKeyInfo
       FROM PKIX1Explicit-2009 -- in [RFC5912]
       { iso(1) identified-organization(3) dod(6) internet(1)
        security(5) mechanisms(5) pkix(7) id-mod(0)
        id-mod-pkix1-explicit-02(51) } ;

   ct-ResourceDistributionConsensus CONTENT-TYPE ::=
    { TYPE ResourceDistributionConsensus
     IDENTIFIED BY id-ct-resourceDistributionConsensus }

   id-ct-resourceDistributionConsensus OBJECT IDENTIFIER ::=
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs-9(9) id-smime(16) id-ct(1) X5 }

   URI   ::= IA5String
   TaName  ::= IA5String
   Filename ::= IA5String

   ResourceDistributionConsensus ::= SEQUENCE {
    taDetails SEQUENCE (SIZE(1..MAX)) OF taDetail,
    otherTaDetails SEQUENCE (SIZE(1..MAX)) OF taDetail,
    bpkiTaKey SubjectPublicKeyInfo,
    uriRdrBase URI,
    bpkiTaFilename Filename,
    rdsFilename Filename }

   taDetail ::= SEQUENCE {
    taName  TaName,
    taKey   SEQUENCE (SIZE(1..MAX)) OF SubjectPublicKeyInfo }

   END
```

This signed object is signed by an EE certificate issued by the TA directly.

A non-normative guideline for naming this object is that the filename be a value derived from the public key part of the entity’s key pair, using the algorithm described for CRLs in Section 2.2 of [RFC6481].

The filename extension of ".rdc" MUST be used to denote the object as an RDC object.

6. Protocol Walkthrough

In this section we will give a walkthrough of how participating TAs can use RDC, RDS and RDE objects to convey their consensus about INR constraints, and how these messages can be validated by constraint validators.

This section will follow the same subsection structure as was used by the "Requirements" section.

6.1. Signing Outside of RPKI Repository

Participating TAs each need to set up their Resource Distribution Repository (RDR), used to distribute RDOs. This involves reserving a base HTTPS URL for the RDR, and ensuring that they have the infrastructure in place that allows them to publish RDOs objects here such that they will be publicly available.

Furthermore, each participant MUST issue, or reuse, a BPKI TA certificate. These certificates are similar to the BPKI TA certificates used in [RFC8183] [RFC8181] and [RFC6492].

The BPKI TA certificate MUST be published under the RDR base URI. The filename must not collide with any other filenames used for RDS or RDE objects, and this filename MUST be used as the "bpkiTaFilename" in the TA’s RDC object so that constraint validators can find it.

The BPKI TA private key MUST be used to sign EE certificates for use in the CMS for RDS and RDE objects issued by this participant. These EE certificates MUST use single-use key pairs, and MUST have a validity period that ensures that the object remains valid for so long as the participant intends that the associated objects be used in constraint validation. The specific validity period to use is a policy matter for the TAs participating in the consensus group.

6.2. Initiating Participants and Resource Distribution

6.2.1. Participant Set

Consensus about constraints is asserted among a group of participating TAs. Before signing any statements about the distribution of INRs, the participants MUST first all agree which TAs form this group.

A participating TA is formally described in the "taDetail" element of an RDC object. This element consists of a "taName" and a "taKey" element, where the latter is a sequence of one or more "SubjectPublicKeyInfo" elements for each TA key used by a participant. (While a TA will generally have only one TA key at a given point in time, it is possible for a TA to have multiple keys operating concurrently, in order to facilitate key rollover. See [RFC9691] for details on an in-band key rollover process.)

The full participant group is described in the "taDetails" element of the RDC objects, as a sequence of "taDetail" elements. These elements MUST be ordered by lexical order of their "taName".

6.2.2. Initial Resource Distribution State Objects

The participants then need to come to agreement on the set of resources for which each participant is authoritative on an agreed date. The way in which this distribution is initially determined is a local policy matter for the participants, and out of scope for the purposes of this document.

Once this is done, each participant generates an RDS object as follows:

* The "version" element MUST be 1.
* The "date" element MUST contain the agreed initial date, per the first paragraph of this section.
* The "previousRDS" element MUST NOT be present.
* The "urlPrefix" element MUST reflect the base name that the participant will use when publishing RDE objects in its RDR.
* The "rdoIndex" element MUST NOT be present.
* The "delegations" element MUST be present, containing the mapping of INRs to each participant. Delegations MUST be disjoint among participants.

Each participant MUST publish their RDS object under their RDR using a filename of their choosing. This name MUST be used as the "rdsFilename" in the initial RDC object that the participate signs (see next subsection).

A non-normative guideline is that this file be called "current.rds", so as to allow new RDS objects to be published under the same name without requiring adjustments to the RDC object. When a new RDS object is published, the previous RDS object can be republished under a new name, and referred to by way of the "previousRDS" element in the new RDS object.

6.2.3. Initial Resource Distribution Consensus Objects

Once a TA has published the initial RDS object, it may publish the RDC signed object using the details from the previous sections.

The TAs do not need to coordinate as to the timing of the publication of RDS or RDC objects, but the additional validation provided by way of this specification does not take effect unless each configured TA (or each except for one) that is listed as participating in the consensus is publishing those objects.

A participant MUST NOT publish more than one RDC object at a given time.

6.2.4. Validation of Participant Group

Constraint validation may be performed by general purpose RP software that is also used for validation of the RPKI tree, or by specialised software that seeks to validate the constraints applicable to each RPKI TA.

Constraint Validators can be configured with TALs for the participants its operator wants to consider. Constraint Validators MUST validate the TA certificate for each TAL, as well as its manifest and CRL, in order to download and validate each TA’s RDC object.

A consensus group exists where one or more TAs are publishing RDC objects with matching taDetails and otherTaDetails fields, in that each has entries for a given name and a common TA key, and the set of common TA keys from the taDetails field is equal to or a superset of the set of configured TA keys. The RDC objects in the set for which the most configured TAs are available are referred to as the relevant RDC objects.

If more than one such consensus group is available, the Constraint Validator MUST select the one for which it has the most configured TAs, and MUST treat the rest as if they do not exist. This consensus group is referred to as the selected consensus group. If no single consensus group has the most configured TAs (i.e. if there is a tie for most configured TAs among the available groups), the Constraint Validator MUST proceed as though no consensus group exists. If no consensus group exists, constraint validation cannot proceed.

If the Constraint Validator is configured with a TA key that was not found in the selected consensus group, then the only constraint on that TA is that it MUST NOT be allowed to use any INRs claimed by the selected consensus group.

If the Constraint Validator does not have a TAL for one or more participants in the selected consensus group, then it is RECOMMENDED that the missing TALs are added so that constraint validation can be performed optimally.

If a Constraint Validator is configured with TALs for only a subset of the TAs listed in the taDetails fields of the RDCs of each TA from the selected consensus group, it MAY continue to perform constraint validation, albeit using modified procedures as described in the applicable sections below.

The selected consensus group may refer to a TA in the common taDetails fields for which the Constraint Validator has a configured TAL, but which is not publishing a valid RDC object. If this is the case, then the Constraint Validator SHOULD continue to perform constraint validation on the basis of the selected consensus group, albeit using modified procedures as described in the applicable sections below. This behaviour is required in order to prevent constraint validation being inhibited by the actions of a single participant TA.

On each validation run, Constraint Validators MUST check for new RDC objects for each configured TA. In other words, Constraint Validators MUST NOT rely on the absence of an RDC in a given validation run for the purpose of not checking for that RDC on a subsequent validation run.

6.2.5. Validation of the Resource Distribution

Once the selected consensus group is determined, the Constraint Validator MUST first check whether each configured TA in that group has issued an RDS object that matches one issued by each other TA. The Constraint Validator first retrieves the current object for each TA by concatenating urlPrefix and rdsFilename, verifying it against the BPKI TA certificate retrieved by way of the RDC object, and comparing the version numbers, dates and delegation details for each file. If there is no match, the RP works backwards through the previousRDS links for each TA in order to find a matching set of RDS objects for each TA. Note that the previousRDS links may yield 404 Not Found, if the TA has since removed the previous RDS objects. It is also possible for a previousRDS link to cause a loop, in which case the RP should simply stop attempting to retrieve previousRDS files for that TA.

If it is not possible to find a set of matching RDS objects, one for each TA in the selected consensus group, then the Constraint Validator MUST repeat the above process to find a set for all TAs minus one participant. If the Constraint Validator is able to find such a set, the Constraint Validator SHOULD proceed with a selected consensus group that contains only those TAs present in the smaller set, albeit using modified procedures as described in the applicable sections below. If the Constraint Validator is not able to find such a set, then constraint validation cannot proceed.

Furthermore, the Constraint Validator MUST NOT apply the initial resource distribution until it has processed all further changes (RDEs) that were published by each TA in the selected consensus group since their selected RDS object was published.

6.3. Transfers between Participants

Resources can be transferred from one participant, the current holder, to another. For this, the protocol uses RDE objects. These objects are always signed and published by one participant, and are published only in that participant’s repository.

The filename of an RDE object is determined by concatenating:

* the "urlPrefix" from the relevant RDS object;
* the RDE index number (an integer); and
* the suffix ".cms".

This allows other participants and Constraint Validators to actively poll each participant’s RDR for the appearance of new RDEs.

6.3.1. Transfer Initiation

Transfers of INRs are initiated by the participant that currently holds them.

To do so, the participant MUST create a "TransferInitiation" RDE as follows:

* They generate a locally unique value to use as the "TransferId". These are namespaced by way of the initiating TA, so they are not globally unique. The identifier used in the transfer initiation object then carries through to the remaining objects that relate to that transfer.
* They include a date which SHOULD reflect the moment of initiation.
* They include the recipient's TA name, per the RDC object.
* They include one or more INRs to be transferred.
* They publish the RDE using the next available index number.

The issuing participant MUST NOT publish RDEs that would be invalid.

Other participants, as well as Constraints Validators, observe that this new RDE is published. They MUST validate the RDE is correctly signed by the issuer. The content of the Transfer Initiation RDE is further validated as follows:

* The issuer MUST be the current holder of the INRs in question. Participants are considered the current holder of INRs if they have them by way of the matching RDS state for the selected consensus group, or if they received them through a completed transfer or inclusion (see below) since then and they have not subsequently transferred them to another TA since that point. Participants are also considered the current holder of INRs where the resources are part of an accepted transfer, but the initiator of the relevant transfer is not part of the selected consensus group (i.e. the transfer is implicitly finalised on an attempt by the recipient to initiate a transfer for the resources).

* There MUST NOT be any other unfinished transfer for overlapping INRs. A transfer is unfinished if it was initiated, but no Transfer Finalisation or Transfer Cancellation RDE has been published for it.

If this validation fails, the transfer initiation RDE MUST be rejected (i.e. treated as though it did not exist). The issuing participant SHOULD NOT publish a Transfer Initiation that would be rejected.

If the Transfer Initiation is validated successfully, this does not yet lead to a change in the constraints of participants. That change takes effect once the transfer has been accepted by the recipient as described below. However, if the Constraint Validator's selected consensus group does not include the TA that is nominated as the recipient of the transfer, then it MUST treat the Transfer Initiation as implicitly accepted by the recipient.

6.3.2. Transfer Acceptance

Participants SHOULD monitor the RDR of all other participants for the publication of transfer-related RDEs in which they are nominated as the recipient.

When a participant discovers a valid Transfer Initiation RDE where they are the recipient, they are will need to make a choice whether to accept the transfer or not. If the recipient accepts the transfer, they MUST create a Transfer Accepted RDE as follows:

* They include the TransferId used by the initiator.
* They include a date which SHOULD reflect the moment of acceptance.
* They include the TA name of the initiator, per the RDC object.
* They include the INRs from the Transfer Initiation RDE.
* They publish the RDE using the next available index number.

Other participants, as well as Constraints Validators, observe that this new RDE is published. They MUST validate the RDE is correctly signed by the issuer.

If the Constraint Validator's selected consensus group does not include the TA that is nominated as the initiator of the transfer, then it MUST treat the Transfer Acceptance as implicitly initiated by the initiator. Otherwise, it must verify the transfer as follows:

* There MUST be a matching valid Transfer Initiation object that has the same TransferId, published under the RDR of the initiator. If no such matching object is found, this may be caused by timing issues. The Constraint Validator MUST reject the Transfer Acceptance object (i.e. treat it as if it did not exist) on this validation run if this happens.

* The recipient name in the Transfer Initiation object MUST match that of the participant that signed the Transfer Acceptance object.

* The INRs in the Transfer Acceptance object MUST be identical to the INRs in the Transfer Initiation object.

If the initiator finds through monitoring of their intended recipient's RDR that the recipient published a matching Transfer Acceptance object that is invalid, the initiator MUST issue a Transfer Cancellation RDE.

If a Constraint Validator successfully validates the Transfer Acceptance object, it MUST treat the recipient as also having the transferred resources from that point. Permitting both TAs to speak for the resources for a period of time allows for the recipient TA to create signed objects that correspond to those published by the source TA, avoiding the need for there to be a gap in time during which such objects are unavailable.

The recipient is not allowed to initiate a transfer of these INRs until the transfer is fully completed, per the next phase of this process.

If the recipient is not willing to accept the transfer, then they simply do not publish an RDE accepting it. They SHOULD communicate this to the initiator out-of-band. In this case, the initiator MUST issue a Transfer Cancellation RDE.

6.3.3. Transfer Finalisation and Cancellation

A transfer can be finished in two mutually exclusive ways: finalisation, and cancellation.

Cancellation of the transfer can happen regardless of whether the recipient accepted the transfer. If the recipient accepted the transfer, then they are simply no longer considered as having the relevant resources per constraint validation. The recipient SHOULD monitor for this and remove relevant issued RPKI certificates and objects in the event of cancellation.

The initiator MUST only publish a Transfer Finalisation RDE where they have observed a valid Transfer Acceptance RDE from the recipient. The initiator is not required to publish the finalisation RDE immediately. For example, as a matter of local policy, the two TAs may agree to delay finalisation for a period of time in order to support the creation of matching RPKI signed objects under the recipient's TA.

When other participants and Constraint Validators observe and validate the Transfer Finalisation RDE, they conclude that the resource is now uniquely held by the recipient. Under constraint validation, the issuer is no longer considered to hold the resources. This also means that they cannot transfer these INRs again.

Once a transfer is finalised, it cannot be undone as such. However, a new transfer for the INRs can be initiated by the new holder (i.e. the recipient of the finalised transfer). This can help in scenarios where the recipient and issuer agree that the transfer was cancelled or finalised in error.

6.4. Movements of INRs into or out of the Consensus

Any participant can sign a Resource Inclusion RDE for any resources that are not currently contained in the latest RDS, and that have not not already been the subject of a Resource Inclusion RDE issued by another partcipant.

In addition to this, any participant can sign a Resource Exclusion RDE in order to disclaim its holdership of specific INRs.

Contrary to transfers between participants, these RDEs are signed by one participant only, and take immediate effect. When these RDEs are observed and validated by other participants and Constraint Validators, they MUST shrink or extend the constraints of the issuer with the cited INRs.

6.5. Updated Resource Distribution Set

Periodically, the TAs will need to publish new RDS objects, in order to limit the work required for Constraint Validators that are not caching the results from previous RDS and RDE retrieval operations.

Each participant agrees to a common date and time to be used in the new RDS object.

The new RDS object MUST use the same URL prefix as the previous RDS object, to ensure that RDEs issued after publication of this RDS object can also be found by Constraint Validators that are still relying on the previous RDS object.

The version number in the new RDS object is the next integer after the one from the previously-issued object. Each subsequent RDS object also contains the URL of the previous object, so that while only some TAs have published the new file, the verification process can be applied to the previous RDS objects and intervening RDEs, which will still be published by each TA. As with RDC and RDS objects generally, if a single configured TA is publishing an incorrect or invalid RDS object, validation can still proceed on the basis of the remaining available data.

The new RDS object's resource disposition MUST be equal to that of the previous RDS object with the RDEs from one after the first RDS's rdoIndex up to the last RDE with a date prior to the agreed date of the new RDS object, plus the corresponding RDEs for the other TAs. In principle, this can be used for auditing, but more importantly this allows all participants to arrive at an agreed state for the distribution of the new delegations for each participant independently.

The rdoIndex of the new RDS object MUST be set to the integer value of the last RDE published by the signing participant with a date and time prior to the agreed date and time for the new RDS object.

Then, each participant publishes their new RDS object under the "rdsFilename" used in their RDC object. The previous RDS file MUST be republished under a different name and MUST be referred to by the "previousRDS" in the new RDS object.

Since clients do not fetch previous RDEs, a participant that has initiated an unfinalised transfer as at the time of RDS generation will need to publish an additional transfer initiation object once the new RDS has been published. Similarly, new acceptance RDEs may need to be republished as well. The content of these objects MUST be identical to that of their previously published counterparts.

Constraint Validators that continue from a previous state MAY choose not to use the new RDS file set. However, they MUST be aware that because of the possible republication of prior RDEs, they may see RDEs that are duplicates of RDEs they have already processed. If this occurs, the Constraint Validator should treat the duplicate RDEs as if they do not exist.

Finally, participants may periodically remove old RDS and RDE objects in order to limit the size of the repository. If they do, it is recommended that a public archive of these objects is made available for auditing and research purposes, but note that the constraint validation process does not depend on this.

6.6. Removing Participants

6.6.1. Remove Participant from Consensus Group

In this scenario a participant is removed from the consensus entirely by the other participants and their resources are no longer part of the resource distribution. Essentially this means that the remaining participants no longer cooperate on constraint consensus with the removed participant, but they do not wish to constrain the removed participant's resource claims beyond that it MUST NOT be allowed to

use any INRs claimed by the trusted consensus group.

In order to do this, the remaining participants MUST each publish a new RDC object where the participant is removed from the "taDetails" field, and they MUST each publish an updated RDS objects where all references to the participant are removed.

6.6.2. Remove and Constrain a Participant

In this scenario a participant is removed from the consensus by the other participants, but the remaining participants still wish to constrain resource use by the removed participant.

In order to do this, the remaining participants MUST each publish a new RDC object where the participant is moved from "taDetails" to "otherTaDetails".

When Constraint Validators encounter a TA in "otherTaDetails" for which they have a TAL, they need to use the following modifications to the constraint validation process described in this document:

* They MUST constrain the INRs for it in accordance with the constraints signed by the selected consensus group.
* They MUST disregard any RDEs published by a TA listed in "otherTaDetails".
* The implicit initiation/acceptance/finalisation behaviour applies with respect to such TAs in the same way as for TAs that are part of the consensus from the Constraint Validator’s selected consensus group, but which are not themselves part of that selected consensus group.

The remaining participants MAY publish new RDS objects in which the INRs associated with a TA in "otherTaDetails" are removed. In this case, the consensus group is expressing their shared belief that the removed TA is not to be trusted for any INRs.

In order to undo the above, the remaining participants MUST each publish a new RDC object where the removed participant is moved back from "otherTaDetails" to "taDetails". Any INRs that were no longer associated with that participant can then be transferred back, or included by the participant.

6.7. Adding Participants

To add a new participant into the consensus, they first need to set up their signing infrastructure as described in section 6.1. Furthermore, they SHOULD publish inclusion RDEs for any of their resources that are not in the current RDS for the consensus.

Next, all current participants in the consensus set as well as the new participant MUST publish an updated RDC object that includes the extended set of TAs in the "taDetails" in accordance with the description in section 6.2.1.

If the new participant published inclusion RDEs prior to joining the consensus set, then all participants (including the new participant) SHOULD publish renewed RDS objects as described in section 6.5.

From this point forward the new participant can participate in transfers as described in section 6.3.

It is RECOMMENDED that Constraint Validators that do not have a TAL configured for the newly-added participant to issue a notification to operators prompting them to evaluate whether to add the TAL for the new participant.

Note that Constraint Validators that do not have a TAL for the new participant will treat it as a TA that is not part of the selected consensus group, as described in the relevant sections of this document.

When a Constraint Validator adds a TAL for a current participant, they MUST retrieve the most recent common RDS as described in section 6.5, and then process all subsequent RDEs for all participants in their set of TALs, to rebuild the current resource constraint set as among the participants.

7. Operational Considerations

7.1. Publishing New RDS Objects

A new RDS object published by a TA must effectively be equivalent to the final consensus resource set for the consensus as at the time immediately before its publication. Due to risks associated with race conditions, TAs must coordinate on publication of new RDS objects. For example, the TAs may agree not to publish any RDEs during some window of time, so that the TAs can each generate and publish a new RDS object on the basis of the existing state as at that point.

7.2. Removing Participants

This document describes a number of methods for removing a participant, and the outcome in terms of verifiable intent of the remaining participants. It intentionally does not prescribe which of these is most appropriate, as this is highly dependent on the circumstances and the trust relationship between the remaining and removed participants.

8. Security Considerations

8.1. Adverse Action by a TA

An important requirement of this specification is that a single participant cannot cause constraint validation to fail for the remaining participants.

A participant could cause the formation of the initial consensus group and resource distribution to fail. They could also try to achieve this by removing earlier signed RDC and/or RDS and RDE objects.

A participant can also fail to cooperate in transfers, but in doing so they can only affect transfers that involve them. They cannot successfully transfer resources to themselves without the consent of the participant that is the current holder.

A participant can prevent new updated RDS objects from being accepted. In this case, they can force constraint validators to use an earlier RDS and replay more RDEs than would otherwise be needed.

In each of these cases, the remaining participants can coordinate to remove the participant taking an adverse action from the consensus group, if required.

8.2. Constraint Validation Failures

An RP may want to fall back to standard validation for one or more of its configured TAs in the event of problems with the consensus validation process. Along similar lines, a consensus group may attest to a resource set for an arbitrary TA that causes objects issued by that TA to be treated as if they do not exist, due to overclaim, and an RP may also want to use standard validation for that TA if that occurs. RP developers should offer configuration options accordingly.

8.3. Resource Inclusion

Resource inclusion for resources currently outside the consensus is available to all participating TAs on their own initiative. This is obviously a weaker requirement than that imposed by RDS file processing, where each TA has to agree to the current disposition of resources in its entirety. However, there are a number of considerations that weigh in favour of this approach:

* unused resources tend to be much less of a security concern when compared with used resources;
* requiring approval from each other TA leads to substantial additional complexity; and
* subsequent RDS file publication acts to re-establish the consensus for all resources.

As with adverse action more generally, TAs may act to exclude a TA from a consensus by revoking their existing consensus objects and issuing new ones that exclude that TA.

8.4. Multiple Consensus Groups

Only one consensus group can have effect on any given validation run, but each configured TA is treated as equal when it comes to determining the consensus group that has effect. For example, an RP is configured with five production TAs that publish consensus state (e.g. the RIRs), along with six other TAs that do not. If the six other TAs begin to coordinate on the publication of consensus objects, then the RP will start using that consensus information in preference to that published by the production TAs. RP software should alert users to this possibility, as well as offering configuration options around consensus preference, in order to limit the chance of this sort of unexpected behaviour.

8.5. Other Routing Data Published by Revoked TA

A consensus group may effectively revoke any other TA. Since a given TA may also operate an IRR server, and it is common for RPs to use both RPKI and IRR data when making routing decisions, RPs should consider whether TA revocation by way of consensus information publication should carry through to their use of associated IRR data.

8.6. TAK Objects

For TA revocation to be effective, it will likely be necessary for the TAs still participating in the consensus to monitor for successor keys for the revoked TA, adding those keys to the list of keys for the revoked TA as they are published.

9. General Considerations

This document is intended to align with the emerging consensus in the RIR Governance Requirements (https://www.nro.net/policy/internet-coordination-policy-2/) document that arises from the ICP-2 update process. While the TA constraints functionality is intended to be usable by any group of RPKI TAs, the final specification will be such that the RIRs will be able to implement the TA constraints functionality as well as fulfilling all of the requirements from the governance requirements document.

10. Alternative Solutions

10.1. Single Trust Anchor

One alternative solution to the problem identified in the introduction is to set up a single new TA that issues resource certificates to each of the current TAs. In the RIR context, for example, IANA might operate that role. However, assuming that such an operator can be found in the first place, such a change will generally involve a large amount of work in equipping the new TA operator to carry out their new administrative role. In the IANA context, for example, IANA would have to keep track of inter-RIR transfers, where the volume of transactions is much greater than those related to delegations from IANA to the RIRs.

10.2. Single TA plus peer CAs

A slightly different approach is to have a single TA operator be responsible only for the original delegations to the current TAs, with the current TAs themselves operating as CAs for each other current TA for transfers. The problem with that approach is that either the original recipient of the resources from the single TA has to be involved in every subsequent transfer of those resources, which is an additional administrative burden, or each transferror has to issue a certificate to each transferree, which means that the certificate path can become arbitrarily long, or that additional administrative work is required around certificate maintenance. (See generally https://blog.apnic.net/2020/04/21/rpki-and-trust-anchors/.) (https://blog.apnic.net/2020/04/21/rpki-and-trust-anchors/.))

10.3. Single Synthetic TA Based On Delegated Statistics

Another approach is set out at An experimental single TA for the RIR with restrictions aligned to delegated-nro-latest (https://labs.apnic.net/nro-ta/). This involves deploying a new TA which itself signs the existing operational CA certificates issued by each RIR, but only for the resources that are actually issued to the relevant RIR, so as to limit overclaiming by each RIR. To determine the resource disposition as at a given time, the NRO delegated statistics (https://ftp.ripe.net/pub/stats/ripencc/nro-stats/latest/nro-delegated-stats) file is used, which means that this TA does not have to be run by an RIR or a similarly-situated party, but could be run by anybody, including by operators directly. The fundamental problem with this approach is that it depends on the correctness of the previously-mentioned NRO delegated statistics file, which is unsigned, and for which no verifiable assurances are provided as to its correctness.

11. IANA Considerations

TBD

12. Acknowledgments

TBD

13. Normative References

[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>.

[RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, DOI 10.17487/RFC6480, February 2012, <https://www.rfc-editor.org/info/rfc6480>.

[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>.

[RFC6492] Huston, G., Loomans, R., Ellacott, B., and R. Austein, "A Protocol for Provisioning Resource Certificates", RFC 6492, DOI 10.17487/RFC6492, February 2012, <https://www.rfc-editor.org/info/rfc6492>.

[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>.

[RFC8183] Austein, R., "An Out-of-Band Setup Protocol for Resource Public Key Infrastructure (RPKI) Production Services", RFC 8183, DOI 10.17487/RFC8183, July 2017, <https://www.rfc-editor.org/info/rfc8183>.

[RFC8630] Huston, G., Weiler, S., Michaelson, G., Kent, S., and T. Bruijnzeels, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", RFC 8630, DOI 10.17487/RFC8630, August 2019, <https://www.rfc-editor.org/info/rfc8630>.

[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>.

[RFC9691] Martinez, C., Michaelson, G., Harrison, T., Bruijnzeels, T., and R. Austein, "A Profile for Resource Public Key Infrastructure (RPKI) Trust Anchor Keys (TAKs)", RFC 9691, DOI 10.17487/RFC9691, December 2024, <https://www.rfc-editor.org/info/rfc9691>.

14. Informative References

[RFC8181] Weiler, S., Sonalker, A., and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", RFC 8181, DOI 10.17487/RFC8181, July 2017, <https://www.rfc-editor.org/info/rfc8181>.

Authors' Addresses

Tom Harrison
APNIC
Email: tomh@apnic.net

Tim Bruijnzeels
RIPE NCC
Email: tbruijnzeels@ripe.net

Carlos Martinez-Cagnazzo
LACNIC
Email: carlos@lacnic.net

Mark Kosters
ARIN
Email: markk@arin.net

Yogesh Chadee
AFRINIC
Email: yogesh@afrinic.net