Network Working Group
Internet-Draft
Intended status: Best Current Practice
Expires: 26 December 2025

T. Bruijnzeels
T. de Kock
RIPE NCC
F. Hill
ARIN
T. Harrison
APNIC
J. Snijders
24 June 2025

RPKI Publication Server Best Current Practices
draft-ietf-sidrops-publication-server-bcp-03

Abstract

This document describes best current practices for operating an RFC 8181 RPKI Publication Server and its rsync (RFC 5781) and RRDP (RFC 8182) public repositories.

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 26 December 2025.

Copyright Notice

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

[RFC8181] describes the RPKI Publication Protocol used between RPKI Certification Authorities (CAs) and their Publication Repository server. The server is responsible for handling publication requests sent by the CAs, called Publishers in this context, and ensuring that their data is made available to RPKI Relying Parties (RPs) in (public) rsync and RRDP [RFC8182] publication points.

In this document, we will describe best current practices based on the operational experience of several implementers and operators.

3. Glossary

Term	Description
Publication Server	[RFC8181] Publication Repository server
Publishers	[RFC8181] Publishers (Certification Authorities)
RRDP Server	Public facing [RFC8182] RRDP repository
Rsync Server	Public facing rsync server
rsyncd	Software daemon package running on the Rsync Server
RIR	Regional Internet Registry
NIR	National Internet Registry

Table 1

4. Publication Server

The Publication Server handles the server side of the [RFC8181] Publication Protocol. The Publication Server generates the content for the public-facing RRDP and Rsync Repositories. It is strongly RECOMMENDED that these functions are separated from serving the repository content.

4.1. Self Hosted Publication Server

In general, it is NOT RECOMMENDED to operate a self-hosted Publication Server.

Some organisations that use a self-hosted CA, rather than for example a hosted CA as service provided by their RIR or NIR, also run a self-hosted Publication Server for their CA. In this case, the organisation is responsible for ensuring the availability of the RRDP and rsync content as described in section 5 and 6 of this document.

RPs are expected to make use of cached data from a previous, successful fetch (Section 6 of @!RFC9286). Therefor, short outages on the server side don't need to be cause for immediate concern, provided the server operator restores access availability in a timely fashion (e.g. before objects expire).

However, frequent availability issues with self hosted repositories negatively impact RPs. The greater the number of separate repositories, the greater the chance for negative impact to RPs. Therefore, CAs that act as parents of other CAs are RECOMMENDED to provide a publication service for their child CAs, and CAs with a parent who offers a publication service are RECOMMENDED to use that service, instead of running their own.

For the case of a 'grandchild' CA, where CA1 is a TA, CA2 is a child CA of CA1, and CA3 is a child CA of CA2, there are several options for providing publication service to CA3:

1. RFC 8183 defines a 'referral' mechanism as part of the out-of-band CA setup protocol. If supported by CA1 and CA2, then this simplifies the process of registering CA3 as a direct publication client of CA1.

2. CA1 may support the registration of multiple publishers by CA2, by using the publisher_request/repository_response XML exchange defined in RFC 8183. CA2 would then be able to register a separate publisher on behalf of CA3.

3. CA2 may operate a publication proxy service (per e.g. [rpki-publication-proxy]), which acts as the publication server for CA3. This proxy would set aside part of CA2's namespace at CA1 for the publication of CA3's objects, adjusting and forwarding requests from CA3 to CA1 accordingly.

For options 1 and 2, CAs operating as CA1 should consider the implications of providing direct publication service to CA3 in this way: for example, CA3 may expect publication service technical support from CA1 directly.

4.2. Publication Server as a Service

The Publication Server and repository content have different demands on their availability and reachability. While the repository content MUST be highly available to any RP worldwide, only publishers need to access the Publication Server. Dependent on the specific setup, this may allow for additional access restrictions in this context. For example, the Publication Server can limit access to known source IP addresses or apply rate limits.

If the Publication Server is unavailable for some reason, this will prevent Publishers from making updated RPKI objects available. The most immediate impact of this is that the publisher cannot distribute new issuances or revocations of ROAs, ASPAs or BGPSec Router Certificates for the duration of this outage. Thus, in effect, it cannot signal changes in its routing operations. If the outage persists for an extended period, then RPKI Manifests, CRLs, and Signed Objects might became stale, hampering for example BGP Origin Validation ([RFC6811].

For this reason, the Publication Server MUST have a high availability. Measuring the availability of the Publication Server in a round-trip fashion is recommended by monitoring the publication of objects. Maintenance windows SHOULD be planned and communicated to publishers. This makes publishers aware of the root cause for disruption in the Publication Server that effectively is part of their infrastructure, and helps publishers avoid - if possible - changes in published RPKI objects that are needed during these windows.

4.3. Availability

Short outages of an [RFC8181] publication server will not affect RPs as long as the corresponding RRDP and RSYNC repositories remain available. However, such outages prevent publishers from updating their ROAs and re-issuing their manifests and CRLs in a timely manner.

The propagation time between CA ROA issuance and the ultimate use of resulting VRPs in routers is described in table 2 of [rpki-time-in-flight] and varies between 15 and 95 minutes for the repositories that were the subject of this study. As seen in this study, the delay between signing and publication can be a major contributant to long propogation times.

The potential unavailability of a Publication Server adds to this propagation delay. Publication Servers SHOULD therefore aim for high availability of the [RFC8181] publication protocol.

4.4. Data Loss

Publication Servers MUST aim to minimise data loss in case of severe server outages. If a server restore is needed and a content regression has occured then the server MUST perform an RRDP session reset.

Publishing CAs typically only check in with their Publication Server when they have changes that need to be published. As a result they may not be aware if the server performed a restore and their content regressed to an earlier state. This could result in a number of problems:

* The published ROAs no longer reflect the CA's intentions.
* The CA might not re-issue their Manifest or CRL in time, because they operated under the assumption that the currently published Manifest and CRL have not yet became stale.
* Changes to publishers may not have been persisted. Newly registered publishers may not be present, recently removed publishers may still be present.

Therefore, the Publication Server SHOULD notify publishing CAs about this issue if it occurs, so that a full resynchronisation can be initiated by CAs.

4.5. Publisher Repository Synchronisation

It is RECOMMENDED that publishing CAs always perform a list query as described in section 2.3 of [RFC8181] before sending all their changes using multiple PDUs in a single multi-element query message as described in section 2.2 and section 3.7.1 of [RFC8181]. This way any desynchronisation issue can be resolved at least as soon as the publisher is aware of updates that it needs to publish.

In addition to the above, the publishing CA MAY perform regular planned synchronisation events where it issues an [RFC8181] list query even if it has no new content to publish. For Publication Server that serve a large number of CAs (e.g. 1000s) this operation could become costly from a resource consumption perspective. Unfortunately the [RFC8181] protocol has no proper support for rate limiting. Therefor, publishers SHOULD NOT perform this resynchronisation more frequently than once every 10 minutes unless otherwise agreed with the publication server.

5. RRDP Server

5.1. Distinct Hostnames

It is RECOMMENDED that the public RRDP Server URI uses a different hostname from both the [RFC8181] service_uri used by publishers and the hostname used in rsync URIs (sia_base).

Using a unique hostname will allow the operator to use dedicated infrastructure and/or a Content Delivery Network for its RRDP content without interfering with the other functions.

5.2. Same Origin URIs

Publication Servers need to take note of the normative updates to [RFC8182] in section 3.1 of [RFC9674]. In short this means that all Delta and Snapshot URIs need to use the same host and redirects to other origins are not allowed.

5.3. Endpoint Protection

Repository operators SHOULD use access control to protect the RRDP endpoints. E.g. if the repository operator knows HTTP GET parameters are not in use, then all requests containing GET parameters can be blocked.

5.4. Bandwidth and Data Usage

The bandwidth needed for RRDP evolves and depends on many parameters. These consist of three main groups:

1. RRDP-specific repository properties, such as the size of notification-, delta-, and snapshot files.
2. Properties of the CAs publishing through a particular server, such as the number of updates, number of objects, and size of objects.
3. Relying party behaviour, e.g. using HTTP compression or not, timeouts or minimum transfer speed for downloads, using conditional HTTP requests for notification.xml.

When an RRDP repository server is overloaded, for example, if the bandwidth demands exceed capacity, this causes a negative feedback loop (i.e. the aggregate load increases), and the efficiency of RRDP degrades. For example, when an RP attempts to download one or more delta files, and one fails, it causes them to try to download the snapshot (larger than the sum of the size of the deltas). If this also fails, the RP falls back to rsync. Furthermore, when the RP tries to use RRDP again on the next run, it typically starts by downloading the snapshot.

A Publication Server SHOULD attempt to prevent these issues by closely monitoring performance (e.g. bandwidth, performance on an RP outside their network, unexpected fallback to snapshot). Besides increasing the capacity, we will discuss several other measures to reduce bandwidth demands. Which measures are most effective is situational.

Publication Servers SHOULD support compression. As the RRDP XML and embedded base64 content is highly compressible, this can reduce transferred data by about 50%. Servers SHOULD at least support either deflate or gzip content encoding as described in sections 8.4.1.2 and 8.4.1.3 of [RFC9110] in addition to any other popular compression types that the server can support.

5.5. Content Availability

Publication Servers MUST ensure the high availability of their RRDP repository content.

If possible, it is strongly RECOMMENDED that a Content Delivery Network (CDN) is used to serve the RRDP content. Care MUST be taken to ensure that the Notification File is not cached for longer than 1 minute unless the back-end RRDP Server is unavailable, in which case it is RECOMMENDED that stale files are served.

A CDN will likely cache 404s for files not found on the back-end server. Because of this, the Publication Server SHOULD use randomized, unpredictable paths for Snapshot and Delta Files to avoid the CDN caching such 404s for future updates. Alternatively, the Publication Server can clear the CDN cache for any new files it publishes.

Note that some organisations that run a Publication Server may be able to attain a similar level of availability themselves without the use of a third-party CDN. This document makes no specific recommendations on achieving this, as this is highly dependent on local circumstances and operational preferences.

Also note that small repositories that serve a single CA, and which serve a small amount of data that does not change frequently, may attain high availability using a modest setup. Short downtime would not lead to immediate issues for the CA, provided that the service is restored before their manifest and CRL expire. This may be acceptable to the CA operator, however, because this can negatively impact RPs it is RECOMMENDED that these CAs use a Publication Server that is provided as a service, e.g. by their RIR or NIR, instead if they can.

5.6. Limit Notification File Size

Nowadays, most RPs use conditional requests for notification files, which reduces the traffic for repositories that do not often update relative to the update frequency of RPs. On the other hand, for repositories that update frequently, the content uses the most traffic. For example, for a large repository in January 2024, with a notification file with 144 deltas covering 14 hours, the requests for the notification file used 251GB out of 55.5TB/less than 0.5% of total traffic during a period.

However, for some servers, this ratio may be different. [RFC8182] stipulated that the sum of the size of deltas MUST not exceed the snapshot size to avoid Relying Parties downloading more data than necessary. However, this does not account for the size of the notification file all RPs download. Keeping many deltas present may allow RPs to recover more efficiently if they are significantly out of sync. Still, including _all_ such deltas can also increase the total data transfer because it increases the size of the notification file.

The Notification File size SHOULD be reduced by removing delta files that have been available for a long time to prevent this situation. Because some RPs will only update every 1-2 hours (in 2024), the Publication Server SHOULD include deltas for at least 4 hours.

Furthermore, we RECOMMEND that Publication Servers do not produce Delta Files more frequently than once per minute. A possible approach for this is that the Publication Server SHOULD publish changes at a regular (one-minute) interval. The Publication Server then publishes the updates received from all Publishers in this interval in a single RRDP Delta File.

While, the latter may not reduce the amount of data due to changed objects, this will result in shorter notification files, and will reduce the number of delta files that RPs need to fetch and process.

5.7. Manifest and CRL Update Times

The manifest and CRL nextUpdate time and expiry are determined by the issuing CA rather than the Publication Server.

From the CA’s point of view a longer period used between scheduled Manifest and CRL re-issuance ensures that they will have more time to resolve unforeseen operational issues. Their current RPKI objects would still remain valid. On the other hand, CAs may wish to avoid using excessive periods because it would make them vulnerable to RPKI data replay attacks.

From the Publication Server’s point of view shorter update times result in more data churn due to manifest and CRL refreshes only. As said, the choice is made by the CAs, but in certain setups - particularly hosted RPKI services - it may be possible to tweak the manifest and CRL re-signing timing. One large repository has found that increasing the re-signing cycle from once every 24 hours, to once every 48 hours (still deemed acceptable) reduced the data usage with approximately 50% as most changes in the system are due to re-signing rather than e.g. ROA changes.

5.8. Consistent load-balancing

5.8.1. Notification File Timing

Notification Files MUST NOT be available to RPs before the referenced snapshot and delta files are available.

As a result, when using a load-balancing setup, care SHOULD be taken to ensure that RPs that make multiple subsequent requests receive content from the same node (e.g. consistent hashing). This way, clients view the timeline on one node where the referenced snapshot and delta files are available. Alternatively, publication infrastructure SHOULD ensure a particular ordering of the visibility of the snapshot plus delta and notification file. All nodes should receive the new snapshot and delta files before any node receives the new notification file.

When using a load-balancing setup with multiple backends, each backend MUST provide a consistent view and MUST update more frequently than the typical refresh rate for rsync repositories used by RPs. When these conditions hold, RPs observe the same RRDP session with the serial monotonically increasing. Unfortunately, [RFC8182] does not specify RP behavior if the serial regresses. As a result, some RPs download the snapshot to re-sync if they observe a serial regression.

5.8.2. L4 load-balancing

If an RRDP repository uses L4 load-balancing, some load-balancer implementations will keep connections to a node in the pool that is no longer active (e.g. disabled because of maintenance). Due to HTTP keepalive, requests from an RP (or CDN) may continue to use the disabled node for an extended period. This issue is especially prominent with CDNs that use HTTP proxies internally when connecting to the origin while also load-balancing over multiple proxies. As a result, some requests may use a connection to the disabled server and retrieve stale content, while other connections load data from another server. Depending on the exact configuration (U+2013) for

example, nodes behind the LB may have different RRDP sessions (U+2013) this can lead to an inconsistent RRDP repository.

Because of this issue, we RECOMMEND to (1) limit HTTP keepalive to a short period on the webservers in the pool and (2) limit the number of HTTP requests per connection. When applying these recommendations, this issue is limited (and effectively less impactful when using a CDN due to caching) to a fail-over between RRDP sessions, where clients also risk reading a notification file for which some of the content is unavailable.

6. Rsync Server

In this section, we will elaborate on the following recommendations:

* Use symlinks to provide consistent content
* Use deterministic timestamps for files
* Load balancing and testing

6.1. Consistent Content

A naive implementation of the Rsync Server might change the repository content while RPs transfer files. Even when the repository is consistent from the repository server's point of view, clients may read an inconsistent set of files. Clients may get a combination of newer and older files. This "phantom read" can lead to unpredictable and unreliable results. While modern RPs will treat such inconsistencies as a "Failed Fetch" ([RFC9286]), it is best to avoid this situation since a failed fetch for one repository can cause the rejection of the publication point for a sub-CA when resources change.

One way to ensure that rsyncd serves connected clients (RPs) with a consistent view of the repository is by configuring the rsyncd 'module' path to a path that contains a symlink that the repository-writing process updates for every repository publication.

Following this process, when an update is published:

1. write the complete updated repository into a new directory
2. fix the timestamps of files (see next section)
3. change the symlink to point to the new directory

Multiple implementations implement this behavior ([krill-sync], [rpki-core], [rsyncit], the rpki.apnic.net repositories, a supporting shellscript [rsync-move]).

Because rsyncd resolves this symlink when it chdirs into the module directory when a client connects, any connected RPs can read a consistent state. To limit the amount of disk space a repository uses, a Rsync Server must clean up copies of the repository; this is a trade-off between providing service to slow clients and disk space.

A repository can safely remove old directories when no RP fetching at a reasonable rate is reading that data. Since the last moment an RP can start reading from a copy is when it last "current", the time a client has to read a copy begins when it was last current (c.f. since written).

Empirical data suggests that Rsync Repositories MAY assume it is safe to do so after one hour. We recommend monitoring for "file has vanished" lines in the rsync log file to detect how many clients are affected by this cleanup process.

6.2. Deterministic Timestamps

By default, rsync uses the modification time and file size to determine if it should transfer a file. Therefore, throughout a file's lifetime, the modification time SHOULD NOT change unless the file's content changes.

We RECOMMEND the following deterministic heuristics for objects' timestamps when written to disk. These heuristics assume that a CA is compliant with [RFC9286] and uses "one-time-use" EE certificates:

* For CRLs, use the value of thisUpdate.
* For RPKI Signed Objects, use the CMS signing-time (see ([I-D.spaghetti-sidrops-cms-signing-time]))
* For CA and BGPSec Router Certificates, use the value of notBefore
* For directories, use any constant value.

6.3. Load Balancing and Testing

To increase availability, during both regular maintenance and exceptional situations, a rsync repository that strives for high availability should be deployed on multiple nodes load-balanced by an L4 load-balancer. Because Rsync sessions use a single TCP connection per session, there is no need for consistent load-balancing between multiple rsyncd servers as long as they each provide a consistent view. While it is RECOMMENDED that repositories are updated more frequently than the typical refresh rate for rsync repositories used by RPs client's point of view, breaking this constraint does not cause degraded behavior.

It is RECOMMENDED that the Rsync Server is load tested to ensure that it can handle the requests by all RPs in case they need to fall back from using RRDP (as is currently preferred).

We RECOMMEND serving rsync repositories from local storage so the host operating system can optimally use its I/O cache. Using network storage is NOT RECOMMENDED because it may not benefit from this cache. For example, when using NFS, the operating system cannot cache the directory listing(s) of the repository.

We RECOMMENDED setting the "max connections" to a value that a single node can handle with (1) the available memory and (2) the IO performance available to be able to serve this number of connections in the time RPs allow for rsync to fetch data. Load-testing results show that machine memory is likely the limiting factor for large repositories that are not IO limited.

The number of rsyncd servers needed depends on the number of RPs, their refresh rate, and the "max connections" used. These values are subject to change over time, so we cannot give clear recommendations here except to restate that we RECOMMEND load-testing rsync and re-evaluating these parameters over time.

7. Acknowledgments

This document is the result of many informal discussions between implementers. The authors wish to thank Mike Hollyman for editorial suggestions.

8. Normative References

[I-D.spaghetti-sidrops-cms-signing-time]
Snijders, J. and T. Harrison, "On the use of the CMS signing-time attribute in RPKI Signed Objects", Work in Progress, Internet-Draft, draft-spaghetti-sidrops-cms-signing-time-01, 7 June 2023, <https://datatracker.ietf.org/doc/html/draft-spaghetti-sidrops-cms-signing-time-01>.

[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>.

[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>.

[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>.

[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>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[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>.

[RFC9674] Snijders, J., "Same-Origin Policy for the RPKI Repository Delta Protocol (RRDP)", RFC 9674, DOI 10.17487/RFC9674, December 2024, <https://www.rfc-editor.org/info/rfc9674>.

9. Informative References

[krill-sync]
Bruijnzeels, T., "krill-sync", 2023, <https://github.com/NLnetLabs/krill-sync>.

[rpki-core]
Team, R., "rpki-core", 2023, <https://github.com/RIPE-NCC/rpki-core>.

[rpki-publication-proxy]
APNIC, "rpki-publication-proxy", 2018, <https://github.com/APNIC-net/rpki-publication-proxy>.

[rpki-time-in-flight]
Fontugne, R., Phokeer, A., Pelsser, C., Vermeulen, K., and R. Bush, "RPKI Time-of-Flight: Tracking Delays in the Management, Control, and Data Planes", 2022, <https://www.iijlab.net/en/members/romain/pdf/romain_pam23.pdf>.

[rsync-move]
Snijders, J., "rpki-rsync-move.sh.txt", 2023,
<http://sobornost.net/~job/rpki-rsync-move.sh.txt>.

[rsyncit] Team, R., "rpki-core", 2023,
<https://github.com/RIPE-NCC/rsyncit>.

Authors' Addresses

Tim Bruijnzeels
RIPE NCC
Email: tbruijnzeels@ripe.net

Ties de Kock
RIPE NCC
Email: tdekock@ripe.net

Frank Hill
ARIN
Email: frank@arin.net

Tom Harrison
APNIC
Email: tomh@apnic.net

Job Snijders
Email: job@sobornost.net