Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 14 March 2026

R. Bush
Arrcus, DRL, & IIJ Research
R. Austein
Dragon Research Labs
10 September 2025

The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 2

draft-ietf-sidrops-8210bis-22

Abstract

In order to validate the origin Autonomous Systems (ASes) and Autonomous System relationships behind BGP announcements, routers need a simple but reliable mechanism to receive Resource Public Key Infrastructure (RFC6480) prefix origin data, Router Keys, and ASPA data from a trusted cache. This document describes a protocol to deliver them.

This document describes version 2 of the RPKI-Router protocol. [RFC6810] describes version 0, and [RFC8210] describes version 1. This document is compatible with both.

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 14 March 2026.

Copyright Notice

Table of Contents

1. Introduction

In order to verifiably validate the origin Autonomous Systems (ASes) and AS paths of BGP announcements, routers need a simple but reliable mechanism to receive cryptographically validated Resource Public Key Infrastructure (RPKI) [RFC6480] prefix origin data, Router Keys, and ASPA data from a trusted cache. This document describes a protocol to deliver them. The design is intentionally constrained to be usable on much of the current generation of ISP router platforms.

This specification documents version 2 of the RPKI-RTR protocol. Earlier versions are documented in [RFC6810] and [RFC8210]. Though this version is, of course, preferred, the earlier versions are expected to continue to be productively deployed indefinitely, and Section 7 details how to downgrade from this version to earlier versions as needed in order to interoperate.

Section 3 describes the deployment structure, and Section 4 then presents an operational overview. The binary payloads of the protocol are formally described in Section 5, and the expected Protocol Data Unit (PDU) sequences are described in Section 8. The transport protocol options are described in Section 9. Section 10 details how routers and caches are configured to connect and authenticate. The traditional security and IANA considerations end the document.

The protocol is extensible in order to support new PDUs with new semantics, if deployment experience indicates that they are needed. PDUs are versioned should deployment experience call for change.

1.1. 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.

1.2. Changes from RFC8210

This section summarizes the significant changes between [RFC8210] and the protocol described in this document.

* A new ASPA (Autonomous System Provider Authorization) PDU type (Section 5.12) has been added to support [I-D.ietf-sidrops-aspa-profile].

* Section 11 has been added in order to handle race conditions, by mandating a payload PDU ordering for caches and documenting related client implementation options.

* Language was clarified when multiple caches are configured, and an interesting affect is noted.

* The protocol version number incremented from 1 (one) to 2 (two) and Section 7 on Protocol Version Negotiation has been updated accordingly.

* Limits the maximum size of a PDU to 64k.

2. Glossary

The following terms are used with special meaning.

Global RPKI: The authoritative data of the RPKI are published in a distributed set of servers at the IANA, Regional Internet Registries (RIRs), National Internet Registries (NIRs), and ISPs; see [RFC6481].

CA: The authoritative data of the RPKI are meant to be published by a distributed set of Certification Authorities (CAs) at the IANA, RIRs, NIRs, and ISPs (see [RFC6481]).

Cache: A Cache, AKA Relying Party Cache, is a coalesced copy of the

published Global RPKI data, periodically fetched or refreshed, directly or indirectly, using the rsync protocol [RFC5781] or some successor. Relying Party software is used to gather and validate the distributed data of the RPKI into a cache. Trusting this cache further is a matter between the provider of the cache and a Relying Party.

Serial Number: "Serial Number" is a 32-bit strictly increasing unsigned integer which wraps from 2^32-1 to 0. See [RFC1982] on DNS Serial Number Arithmetic for too much detail on serial number wrapping. The Serial Number denotes the logical version of a cache. A cache increments the value when it successfully updates its data from a parent cache or from primary RPKI data. While a cache is receiving updates, new incoming data and implicit deletes are associated with the new Serial Number but MUST NOT be sent until the fetch is complete. A Serial Number is not commensurate between different caches or different protocol versions, nor need it be maintained across resets of the cache server.

Session ID: When a cache server starts a new Sequence Number space, (which might be caused by, for example, restart with loss of data) it generates a new Session ID to uniquely identify the instance of the cache and to bind it to the sequence of Serial Numbers that the cache instance generates. This allows a router to resume a session after a transport connection failure without invalidating the router’s data store; as it is assured that the Serial Numbers it uses are commensurate with those of the cache.

Payload PDU: A payload PDU is a protocol message which contains data for use by the router, as opposed to a PDU which conveys the control mechanisms of this protocol. IPvX Prefixes, Router Keys, ASPA are examples of payload PDUs.

3. Deployment Structure

Deployment of the RPKI to reach routers has a three-level structure as follows:

Global RPKI: The authoritative data of the RPKI are published in a distributed set of servers at the IANA, RIRs, NIRs, and ISPs (see [RFC6481]).

Local Caches: Local caches are a local set of one or more collected and verified caches of RPKI data. A Relying Party, e.g., router or other client, MUST have a trust relationship with, and a trusted transport channel to, any cache(s) it uses.

Routers: A router fetches data from a local cache using the protocol

described in this document. It is said to be a client of the cache. There MAY be mechanisms for the router to assure itself of the authenticity of the cache and to authenticate itself to the cache (see Section 9).

4. Operational Overview

A router establishes and keeps open a transport connection to one or more caches with which it has client/server relationships. It is configured with a semi-ordered list of caches and establishes a connection to the most preferred cache, or set of caches with that same priority, which accept the connections.

The router MUST choose the most preferred, by configuration, cache or set of caches so that the operator may control load on their caches and the Global RPKI.

A Validated ROA Payload (VRP, see [RFC6811]) is effective if it is in the fetched set from any of the currently preferred caches. Therefore, a VRP takes effect on the router when the first cache serves that VRP, and the VRP is in effect until the last cache withdraws that VRP. Thus, in a global sense, the effect of a VRP announcement propagates more quickly than a withdraw.

Periodically, the router sends a Serial Query to the cache specifying the most recent Serial Number for which it has received data from that cache, i.e., the router’s current Serial Number for that cache, in the form of a Serial Query. When a router establishes a new session with a cache or wishes to reset a current relationship, it sends a Reset Query.

The cache responds to the Serial Query with all data changes which took place since the given Serial Number. This may be the null set, in which case the End of Data PDU (Section 5.8) is still sent. Note that the Serial Number comparison used to determine "since the given Serial Number" MUST take wrap-around into account; see [RFC1982].

When the router receives an End of Data PDU, it has received all current data from the cache. It then sets its current Serial Number for that cache to that of the Serial Number in the received End of Data PDU.

When the cache updates its database, it sends a Notify PDU to every currently connected router. This is a hint that now would be a good time for the router to poll for an update, but it is only a hint. The protocol requires the router to poll for updates periodically in any case.

Strictly speaking, a router could track a cache simply by asking for a complete data set every time it updates, but this would be very inefficient. The Serial-Number-based incremental update mechanism allows an efficient transfer of just the data records which have changed since the last update. As with any update protocol based on incremental transfers, the router must be prepared to fall back to a full transfer if for any reason the cache is unable to provide the necessary incremental data. Unlike some incremental transfer protocols, this protocol requires the router to make an explicit request to start the fallback process; this is deliberate, as the cache has no way of knowing whether the router has also established sessions with other caches that may be able to provide better service.

As cache servers must evaluate signed objects (see [RFC6480]) with time dependent validity periods, servers' clocks MUST be correct within a tolerance of an hour.

Barring errors, transport connections remain up as long as the cache and router remain up and the router is not reconfigured to no longer use the cache.

Should a transport connection be lost for unknown reasons, the router SHOULD try to reestablish one; being careful to not abuse the cache with two many failed requests.

5. Protocol Data Units (PDUs)

The exchanges between the cache and the router are sequences of exchanges of the following PDUs according to the rules described in Section 8.

Reserved fields (marked "zero" in PDU diagrams) MUST be zero on transmission and MUST be ignored on receipt.

5.1. Fields of a PDU

PDUs contain the following data elements:

Protocol Version: An 8-bit unsigned integer, currently 2, denoting the version of this protocol.

PDU Type: An 8-bit unsigned integer, denoting the type of the PDU, e.g., Type 4, IPv4 Prefix.

Serial Number: A 32-bit unsigned integer serializing the RPKI cache

epoch when this set of PDUs was received from an upstream cache server or gathered from the Global RPKI. A cache increments its Serial Number when completing a validated update from a parent cache or the Global RPKI.

Session ID: A 16-bit unsigned integer. When a cache server is [re]started (i.e. its data are not a continuation of the previous data) it generates a new Session ID to identify the instance of the cache and to bind it to the sequence of Serial Numbers that cache instance will generate. This allows the router to restart a failed session knowing that the Serial Number it is using is commensurate with that of the cache. If, at any time after the protocol version has been negotiated (Section 7), either the router or the cache finds that the value of the Session ID is not the same as the other's, the party which detects the mismatch MUST immediately terminate the session with an Error Report PDU with code 0 ("Corrupt Data"), and the router MUST flush all data learned from that cache.

Note that sessions are specific to a particular protocol version. That is, if a cache server which supports multiple versions of this protocol happens to use the same Session ID value for multiple protocol versions, and further happens to use the same Serial Number values for two or more sessions using the same Session ID but different Protocol Version values, the Serial Numbers are not commensurate. The full test for whether Serial Numbers are commensurate requires comparing Protocol Version, Session ID, and Serial Number. To reduce the risk of confusion, cache servers SHOULD NOT use the same Session ID across multiple protocol versions, but even if they do, routers MUST treat sessions with different Protocol Version fields as separate sessions even if they do happen to have the same Session ID.

Should a cache erroneously reuse a Session ID so that a router does not realize that the session has changed (old Session ID and new Session ID have the same numeric value), the router may become confused as to the content of the cache. The time it takes the router to discover this will depend on whether the Serial Numbers are also reused. If the Serial Numbers in the old and new sessions are different enough, the cache will respond to the router's Serial Query with a Cache Reset, which will solve the problem. If, however, the Serial Numbers are close, the cache may respond with a Cache Response, which may not be enough to bring the router into sync. In such cases, it's likely but not certain that the router will detect some discrepancy between the state that the cache expects and its own state. For example, the Cache Response may tell the router to drop a record which the router does not hold or may tell the router to add a record which the

router already has. In such cases, a router will detect the error and reset the session. The one case in which the router may stay out of sync is when nothing in the Cache Response contradicts any data currently held by the router.

Using persistent storage for the Session ID or a clock-based scheme for generating Session IDs should avoid the risk of Session ID collisions.

The Session ID might be a pseudorandom value, a strictly increasing value if the cache has reliable storage, et cetera. A seconds-since-epoch timestamp value such as the low order 16 bits of unsigned integer seconds since 1970-01-01T00:00:00Z ignoring leap seconds might make a good Session ID value.

Length: A 32-bit unsigned integer which has as its value the count of the octets in the entire PDU, including the 8 octets of header which includes the length field. This length MUST NOT exceed 65,535 octets. Note that BGP speakers already need the capability to handle messages of this size, see [RFC8654].

Flags: An 8-bit binary field, with the lowest-order bit being 1 for an announcement and 0 for a withdrawal. For a Prefix PDU (IPv4 or IPv6), the announce/withdraw flag indicates whether this PDU announces a new right to announce the prefix or withdraws a previously announced right; a withdraw effectively deletes one previously announced Prefix PDU with the exact same Prefix, Length, Max-Len, and Autonomous System Number (AS).

Similarly, for a Router Key PDU, the flag indicates whether this PDU announces a new Router Key or deletes a previously announced Router Key PDU with the exact same AS Number, subjectKeyIdentifier, and subjectPublicKeyInfo.

Similarly, for an ASPA PDU, the flag indicates a new or replacement mapping of the specified Customer AS to a set of Provider ASes or the removal of the existant mapping.

The remaining bits in the Flags field are reserved for future use.

Prefix Length: An 8-bit unsigned integer denoting the shortest prefix allowed by the Prefix element.

Max Length: An 8-bit unsigned integer denoting the longest prefix allowed by the Prefix element. This MUST NOT be less than the Prefix Length element.

Prefix: The IPv4 or IPv6 prefix of the ROA.

Autonomous System Number: A 32-bit unsigned integer representing an AS allowed to announce a prefix, associated with a Router Key, or ASPA Customer or Provider.

Subject Key Identifier: The 20-octet Subject Key Identifier (SKI) value of a router key, as described in [RFC6487].

Subject Public Key Info: A variable length field holding a Router Key's subjectPublicKeyInfo value, as described in [RFC8608]. This is the full ASN.1 DER encoding of the subjectPublicKeyInfo, including the ASN.1 tag and length values of the subjectPublicKeyInfo SEQUENCE.

Refresh Interval: A 32-bit interval in seconds between normal cache polls. See Section 6.

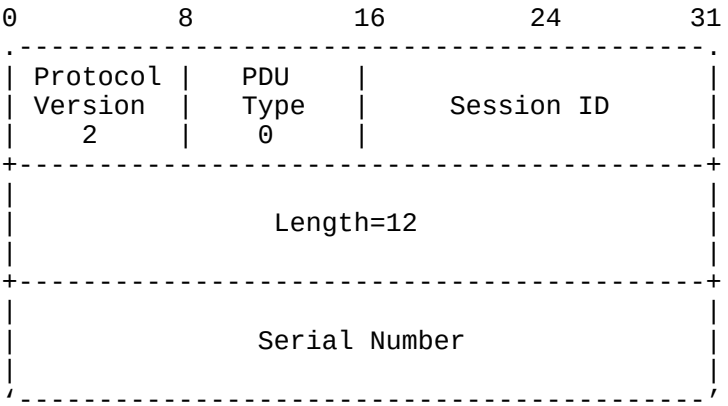
Retry Interval: A 32-bit interval in seconds between cache poll retries after a failed cache poll. See Section 6.

Expire Interval: A 32-bit interval in seconds during which data fetched from a cache remains valid in the absence of a successful subsequent cache poll. See Section 6.

Customer Autonomous System Number: The 32-bit AS number of the Autonomous System that authorizes the upstream providers listed in the Provider Autonomous System list to propagate prefixes to other ASes.

Provider Autonomous System Numbers: The set of 32-bit AS numbers authorized to propagate prefixes which were received from the customer AS.

5.2. Serial Notify



The cache notifies the router that the cache has new data.

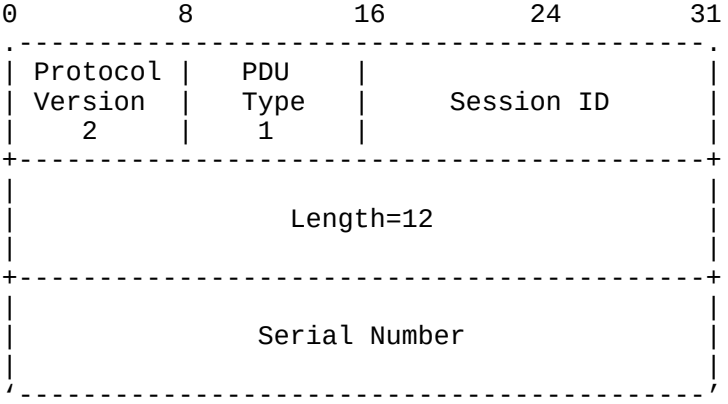
The Session ID reassures the router that the Serial Numbers are commensurate, i.e., the cache session has not been changed.

Upon receipt of a Serial Notify PDU, the router MAY issue an immediate Serial Query (Section 5.3) or Reset Query (Section 5.4) without waiting for the Refresh Interval timer (see Section 6) to expire.

Serial Notify is the only message that the cache MAY send that is not in response to a message from the router.

If the router receives a Serial Notify PDU during the initial startup period where the router and cache are still negotiating to agree on a protocol version, the router MUST simply ignore the Serial Notify PDU, even if the Serial Notify PDU is for an unexpected protocol version. See Section 7 for details.

5.3. Serial Query



The router sends a Serial Query to ask the cache for all announcements and withdrawals which have occurred since the Serial Number specified in the Serial Query.

The cache replies to this query with a Cache Response PDU (Section 5.5) if the cache has a (possibly null) record of the changes since the Serial Number specified by the router, followed by zero or more payload PDUs and an End Of Data PDU (Section 5.8).

When replying to a Serial Query, the cache MUST return the minimum set of changes needed to bring the router into sync with the cache. That is, if a particular prefix, Router Key, or ASPA underwent

multiple changes between the Serial Number specified by the router and the cache's current Serial Number, the cache MUST merge those changes to present the simplest possible view of those changes to the router. In general, this means that, for any particular prefix/AS, Router Key. or ASPA/Customer, the data stream will include at most one withdrawal followed by at most one announcement, and if all of the changes cancel out, the data stream will not mention the prefix/AS, Router Key, or ASPA/Customer at all.

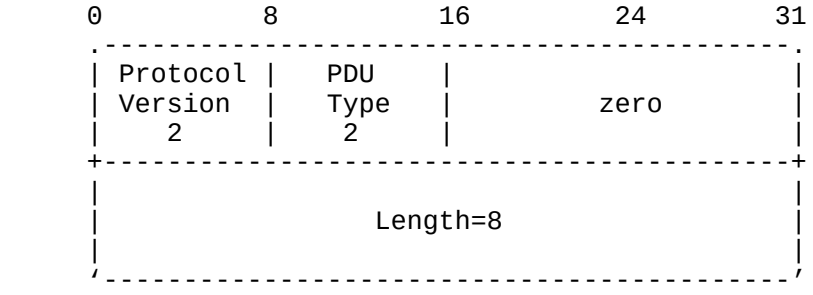
In the data responding to a Serial Query, should the router receive duplicate announcements or withdrawals, the router should raise an Error with Error Code 7, Duplicate Announcement Received.

The rationale for this approach is that the entire purpose of the RPKI-Router protocol is to offload work from the router to the cache, and it should therefore be the cache's job to simplify the change set, thus reducing work for the router.

If the cache does not have the data needed to update the router, perhaps because its records do not go back to the Serial Number in the Serial Query, then it responds with a Cache Reset PDU (Section 5.9).

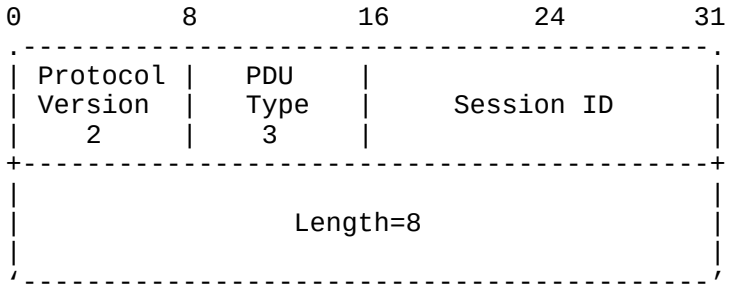
The Session ID tells the cache what instance the router expects to ensure that the Serial Numbers are commensurate, i.e., the cache session has not been changed. If the Session ID does not match, the cache MUST respond with a Cache Reset.

5.4. Reset Query



The router tells the cache that it wants to receive the total active, current, non-withdrawn database. The cache responds with a Cache Response PDU (Section 5.5), followed by zero or more payload PDUs and an End of Data PDU (Section 5.8).

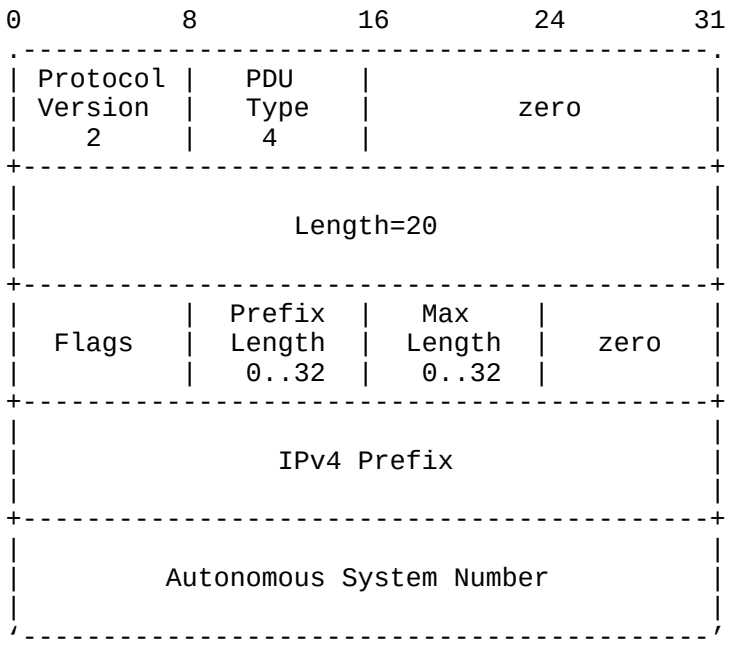
5.5. Cache Response



The cache responds to queries with zero or more payload PDUs. When replying to a Serial Query (Section 5.3), the cache sends the set of announcements and withdrawals necessary to bring the router's state current with all changes that have occurred since the Serial Number sent by the client router. When replying to a Reset Query (Section 5.4), the cache sends the set of all data records it has; in this case, the announce/withdraw field in the payload PDUs MUST have the value 1 (announce).

In response to a Reset Query, the new value of the Session ID tells the router the instance of the cache session for future confirmation. In response to a Serial Query, the Session ID being the same reassures the router that the Serial Numbers are commensurate, i.e., the cache session has not been changed.

5.6. IPv4 Prefix



This PDU carries a VRP for an IPv4 ROA [RFC6811].

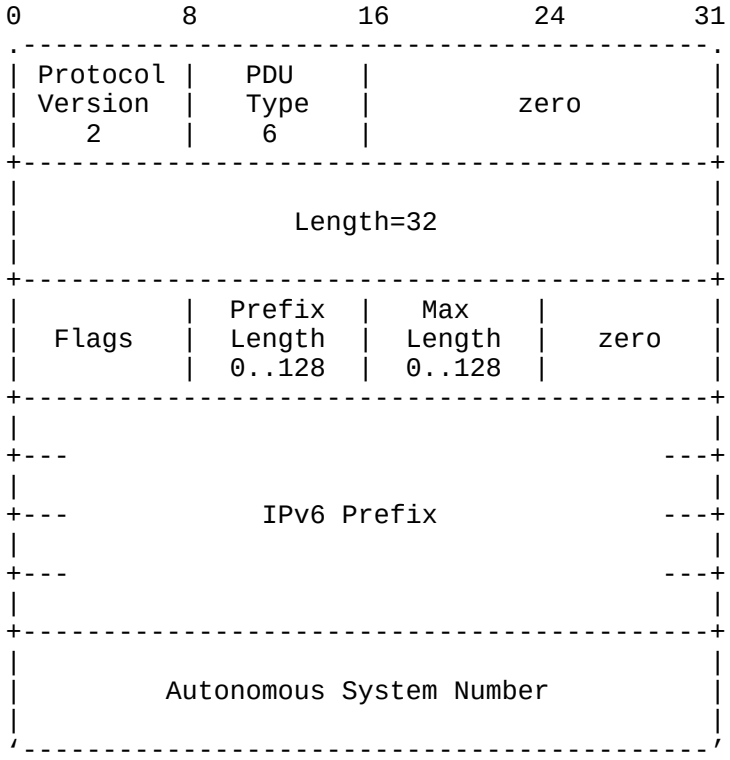
The lowest-order bit of the Flags field is 1 for an announcement and 0 for a withdrawal.

In the RPKI, there is an actual need for what might appear to a router as identical IPvX PDUs. This can occur when an upstream certificate is being reissued or there is an address ownership transfer up the validation chain. The ROA would be identical in the router sense, i.e., have the same {Prefix, Len, Max-Len, AS}, but it would have a different validation path in the RPKI. This is important to the RPKI but not to the router.

The cache server MUST ensure that it has told the router client to have one and only one IPvX VRP for a unique {Prefix, Len, Max-Len, AS} at any one point in time. Should the router client receive an IPvX VRP with a {Prefix, Len, Max-Len, AS} identical to one it already has active, it SHOULD raise a Duplicate Announcement Received error.

The cache MUST merge announce/withdraw ROAs for the same {Prefix, Len, Max-Len, AS} into the minimal (or no) VRP to update the router to to the desired state.

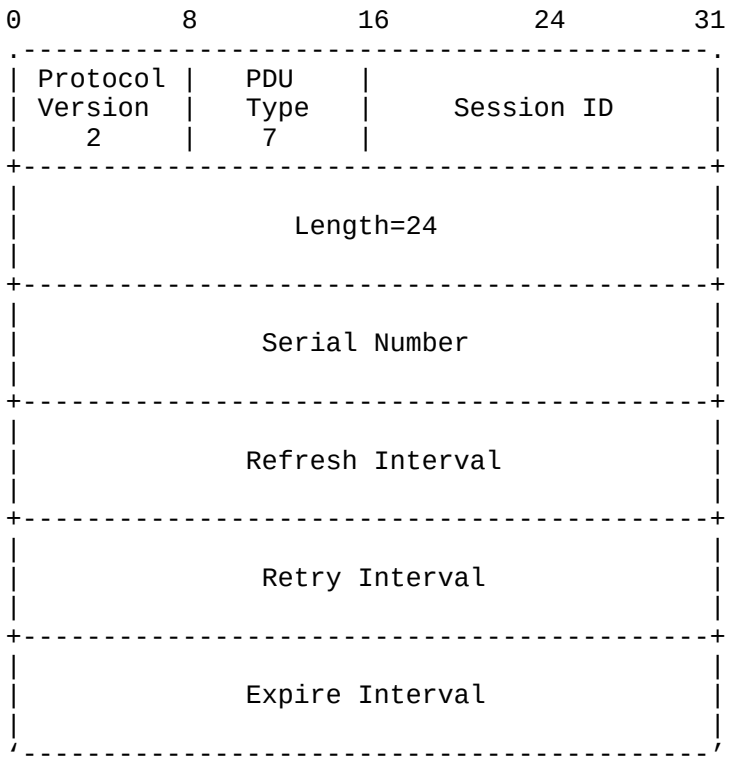
5.7. IPv6 Prefix



This PDU carries a VRP for an IPv6 ROA [RFC6811].

Analogous to the IPv4 Prefix PDU, it has 96 more bits and no magic.

5.8. End of Data



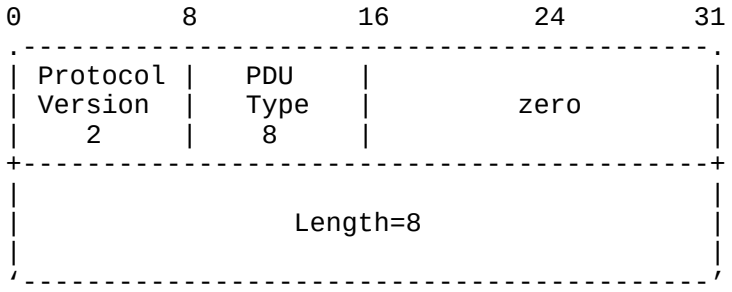
A cache sends an End of Data record to tell the router it has no more data for the request.

The Session ID and Protocol Version MUST be the same as that of the corresponding Cache Response which began the (possibly null) sequence of payload PDUs.

The Refresh Interval, Retry Interval, and Expire Interval are all 32-bit elapsed times measured in seconds. They express the timing parameters which the cache expects the router to use in deciding when to send subsequent Serial Query or Reset Query PDUs to the cache. See Section 6 for an explanation of the use and the range of allowed values for these parameters.

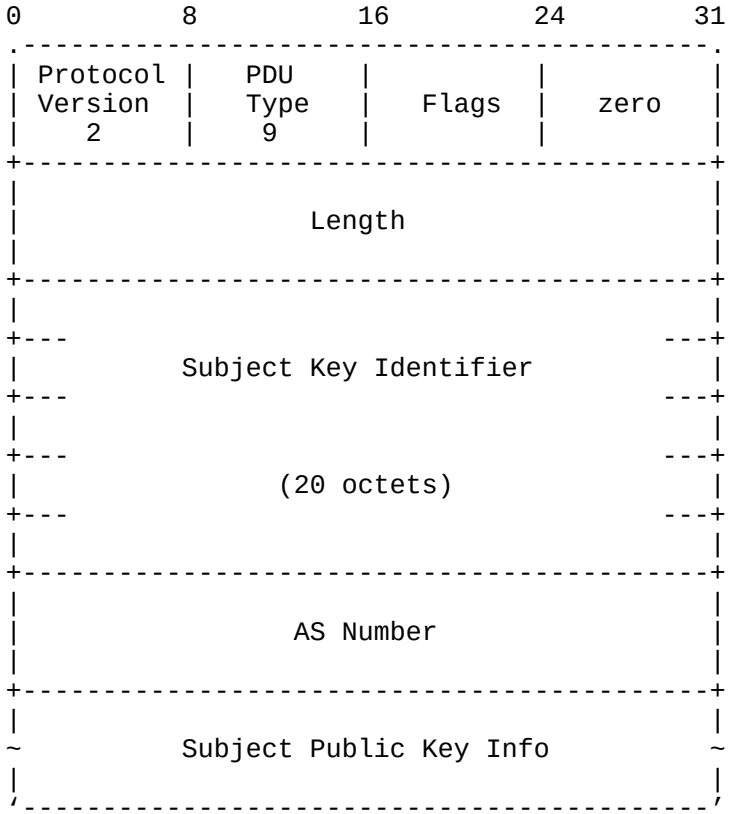
Note that the End of Data PDU changed significantly between versions 0 and 1. The Version 2 End of Data PDU is the same as that of Version 1.

5.9. Cache Reset



The cache sends a Cache Reset PDU in response to a Serial Query in order to inform the router that the cache cannot provide an incremental update starting from the Serial Number specified by the router. The router must decide whether to issue a Reset Query or perhaps switch to a different cache.

5.10. Router Key



The Router Key PDU transports they payload of a Router Key [RFC8635].

The lowest-order bit of the Flags field is 1 for an announcement and 0 for a withdrawal.

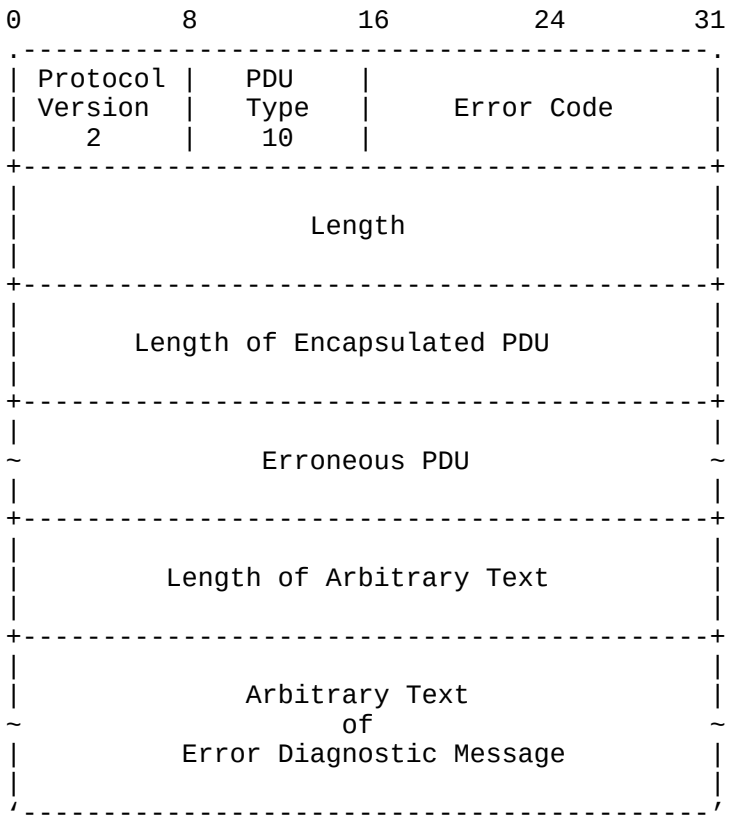
The cache server MUST ensure that it has told the router client to have one and only one Router Key PDU for a unique {SKI, AS, Subject Public Key} at any one point in time. Should the router client receive a Router Key PDU with a {SKI, AS, Subject Public Key} identical to one it already has active, it SHOULD raise a Duplicate Announcement Received error.

Note that a particular AS may appear in multiple Router Key PDUs with different Subject Public Key values, while a particular Subject Public Key value may appear in multiple Router Key PDUs with different ASes. In the interest of keeping the announcement and withdrawal semantics as simple as possible for the router, this protocol makes no attempt to compress either of these cases.

Also note that it is possible, albeit very unlikely, for multiple distinct Subject Public Key values to hash to the same SKI. For this reason, implementations MUST compare Subject Public Key values as well as SKIs when detecting duplicate PDUs.

As the Subject Public Key Info is a variable length field, it must be decoded to determine where the PDU terminates.

5.11. Error Report



This PDU is used by either party to report an error to the other.

Error reports are only sent as responses to other PDUs, not to report errors in Error Report PDUs.

Error codes are described in Section 12.

The Erroneous PDU field is a binary copy of the PDU causing the error condition, including all fields.

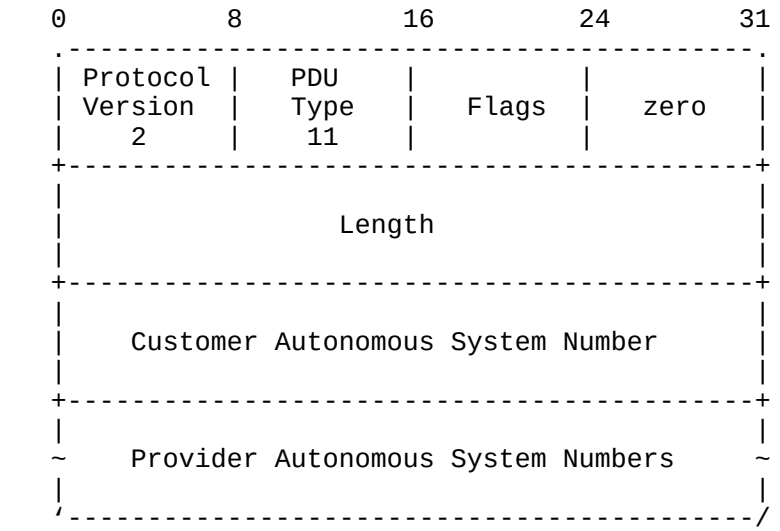
If the error is generic (e.g., "Internal Error") and not associated with the PDU to which it is responding, the Erroneous PDU field MUST be empty and the Length of Encapsulated PDU field MUST be zero.

An Error Report PDU MUST NOT be sent for an Error Report PDU. If an erroneous Error Report PDU is received, the session SHOULD be dropped.

If the entire erroneous PDU will not fit in the Erroneous PDU field, it MUST be truncated. At a minimum, the first four octets MUST be included. Beware any attempts to parse an Erroneous PDU.

The Arbitrary Text field is optional; if not present, the Length of Arbitrary text field MUST be zero. If Arbitrary Text is present, it MUST be a string in UTF-8 encoding (see [RFC3629]) in the Queen's English.

5.12. ASPA PDU



The ASPA PDU supports [I-D.ietf-sidrops-aspa-profile].

The Customer Autonomous System Number is the 32-bit Autonomous System Number of the customer which authenticated the ASPA RPKI data.

There are zero or more 32-bit Provider Autonomous System Number fields in increasing numeric order. See [I-D.ietf-sidrops-aspa-profile].

The Flags field is as described in Section 5.

The router MUST see at most one ASPA from a particular cache for a particular Customer Autonomous System Number active at any time. As a number of conditions in the global RPKI may present multiple valid ASPA RPKI records for a single customer to a particular RP cache, this places a burden on the cache to form the union of multiple ASPA records it has received from the global RPKI into one ASPA PDU.

Receipt of an ASPA PDU announcement (announce/withdraw flag == 1) when the router already has an ASPA PDU with the same Customer Autonomous System Number from that cache replaces the previous one. The cache MUST deliver the complete data of an ASPA record in a single ASPA PDU.

For the ASPA PDU, the announce/withdraw Flag is set to 1 to indicate either the announcement of a new ASPA record or a replacement for a previously announced record from that cache with the same Customer Autonomous System Number. For an announcement, the PDU MUST contain at least one Provider Autonomous System Number or an Error PDU with code 9, ASPA Provider List Error is returned.

If the announce/withdraw flag is set to 0, the entire ASPA record from that cache for that Customer AS MUST be removed from the router. In this case, the customer AS of the ASPA record MUST be provided, there MUST be no Provider list, and the PDU Length MUST be 12.

6. Protocol Timing Parameters

Since the data the cache distributed via the RPKI-Router protocol are retrieved from the Global RPKI system at intervals which are only known to the cache, only the cache can really know how frequently it makes sense for the router to poll the cache, or how long the data are likely to remain valid (or, at least, not significantly changed). For this reason, as well as to allow the cache some control over the load placed on it by its client routers, the End Of Data PDU includes three values that allow the cache to communicate timing parameters to the router:

Refresh Interval: This parameter tells the router how long to wait before next attempting to poll the cache and between subsequent attempts, using a Serial Query or Reset Query PDU. The router SHOULD NOT poll the cache sooner than indicated by this parameter. Note that receipt of a Serial Notify PDU overrides this interval and suggests that the router issue an immediate query without waiting for the Refresh Interval to expire. Countdown for this timer starts upon receipt of the containing End Of Data PDU.

Minimum allowed value: 1 second.

Maximum allowed value: 86400 seconds (1 day).

Recommended default: 3600 seconds (1 hour).

Retry Interval: This parameter tells the router how long to wait

before retrying a failed Serial Query or Reset Query. The router SHOULD NOT retry sooner than indicated by this parameter. Note that a protocol version mismatch overrides this interval: if the router needs to downgrade to a lower protocol version number, it MAY send the first Serial Query or Reset Query immediately. Countdown for this timer starts upon failure of the query and restarts after each subsequent failure until a query succeeds.

Minimum allowed value: 1 second.

Maximum allowed value: 7200 seconds (2 hours).

Recommended default: 600 seconds (10 minutes).

Expire Interval: This parameter tells the router how long it can continue to use the current version of the data while unable to perform a successful subsequent query. The router MUST NOT retain the data past the time indicated by this parameter. Countdown for this timer starts upon receipt of the containing End Of Data PDU.

Minimum allowed value: 600 seconds (10 minutes).

Maximum allowed value: 172800 seconds (2 days).

Recommended default: 7200 seconds (2 hours).

If the router has never issued a successful query against a particular cache, it SHOULD retry periodically using the default Retry Interval, above.

Caches MUST set Expire Interval to a value larger than both the Refresh Interval and the Retry Interval.

7. Protocol Version Negotiation

Once a router has established a transport connection to a cache, it MUST attempt to open a RPKI-Router 'session' by issuing either a Reset Query Section 5.4) or a Serial Query (Section 5.3) with the highest version of this protocol the router implements in the Protocol Version field. If the cache supports that version, it responds with a Cache Response (Section 5.5) of that version and the session is considered open.

If a cache which supports version C receives a query with Protocol Version Q < C, and the cache does not support versions <= Q, the cache MUST send an Error Report (Section 5.11) with Protocol Version C and Error Code 4 ("Unsupported Protocol Version") and disconnect the transport, as negotiation is hopeless.

If a cache which supports version C receives a query with Protocol Version Q < C, and the cache can support version Q, the cache MUST establish the session at protocol version Q, [RFC6810] or [RFC8210], and respond with a Cache Response (Section 5.5) of that Protocol Version, Q, and the RPKI-Rtr session is considered open.

If the the cache which supports C as its highest verion receives a query of version Q > C, the cache MUST send an Error Report with Protocol Version C and Error Code 4. The router SHOULD send another query with a Protocol Version Q with Q == the version C in the Error Report; unless it has already failed at that version, which indicates a fatal error in programming of the cache which SHOULD result in transport termination.

If the router requests Q == 0 and it still fails with the cache responding with an Error Report with Error Code 4, then the router MUST abort the transport connection, as negotiation is hopeless.

In any of the downgraded combinations above, the new features of the higher version will not be available, and all PDUs MUST have the negotiated lower version number in their version fields.

If either party receives a PDU containing an unrecognized Protocol Version (neither 0, 1, nor 2) during this negotiation, it MUST either downgrade to a known version or terminate the connection, with an Error Report PDU unless the received PDU is itself an Error Report PDU.

The router MUST ignore any Serial Notify PDUs it might receive from the cache during this initial startup period, regardless of the Protocol Version field in the Serial Notify PDU. Since Session ID and Serial Number values are specific to a particular protocol version, the values in the notification are not useful to the router. Even if these values were meaningful, the only effect that processing the notification would have would be to trigger exactly the same Reset Query or Serial Query that the router has already sent as part of the not-yet-complete version negotiation process, so there is nothing to be gained by processing Serial Notify PDUs until version negotiation completes.

Caches SHOULD NOT send Serial Notify PDUs before version negotiation completes. Routers, however, MUST handle such notifications (by ignoring them) for backwards compatibility with caches serving protocol version 0.

Once the cache and router have agreed upon a Protocol Version via the negotiation process above, that version is fixed for the life of the session. See Section 5.1 for a discussion of the interaction between Protocol Version and Session ID.

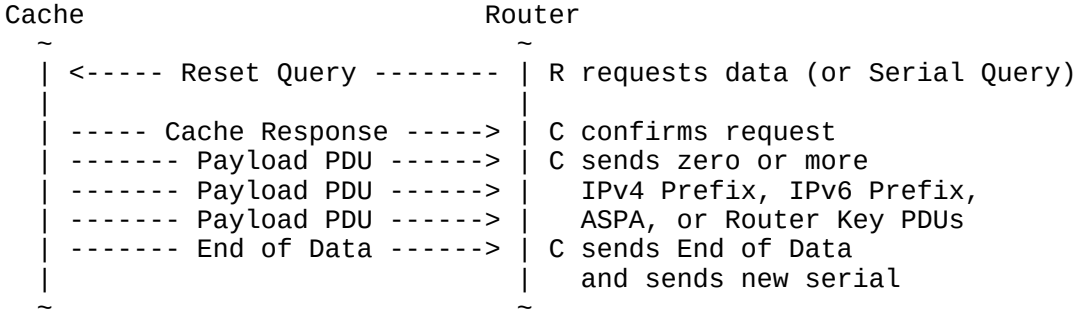
The configured transport security, the negotiated RPKI-Rtr version, etc. MAY NOT be changed once a session has been established. If one side or the other wishes to try a different transport, protocol version, etc. they MUST terminate the transport and restart the entire transport and version negotiation process.

If either party receives a PDU for a different Protocol Version once the above negotiation completes, that party MUST drop the session; unless the PDU containing the unexpected Protocol Version was itself an Error Report PDU, the party dropping the session SHOULD send an Error Report with an error code of 8 ("Unexpected Protocol Version").

8. Protocol Sequences

The sequences of PDU transmissions fall into four conversations as follows:

8.1. Start or Restart



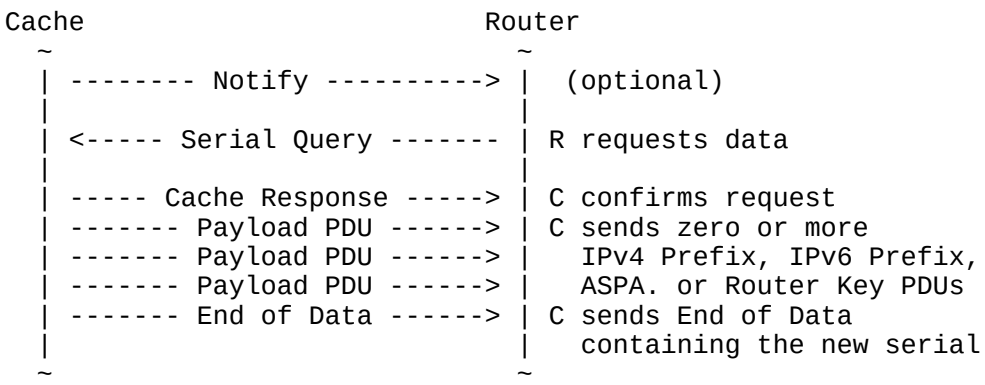
When a transport connection is first established, the router MUST send either a Reset Query or a Serial Query. A Serial Query would be appropriate if the router has unexpired data from a broken session with the same cache and remembers the Session ID of that session, in which case a Serial Query containing the Session ID from the previous session will allow the router to bring itself up to date while ensuring that the Serial Numbers are commensurate and that the router and cache are speaking compatible versions of the protocol. In all other cases, the router lacks the necessary data for fast resynchronization and therefore MUST fall back to a Reset Query.

The Reset Query sequence is also used when the router receives a Cache Reset, chooses a new cache, or fears that it has otherwise lost its way.

See Section 7 for details on version negotiation.

To limit the length of time a cache must keep the data necessary to generate incremental updates, a router MUST send either a Serial Query or a Reset Query periodically. This also acts as a keep-alive at the application layer. See Section 6 for details on the required polling frequency.

8.2. Typical Exchange

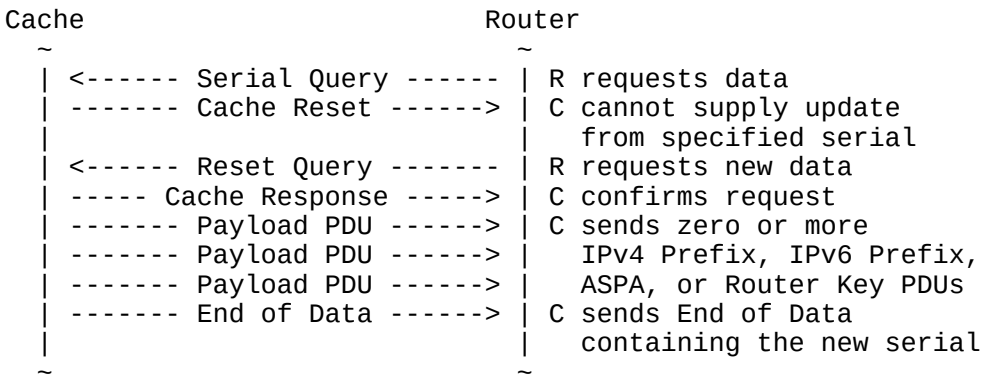


The cache server SHOULD send a Notify PDU with its current Serial Number when the cache's serial changes, with the expectation that the router MAY then issue a Serial Query earlier than it otherwise might. This is analogous to DNS NOTIFY in [RFC1996]. The cache MUST rate-limit Serial Notifies to no more frequently than one per minute.

When the transport layer is up and either a timer has gone off in the router or the cache has sent a Notify PDU, the router queries for new data by sending a Serial Query, and the cache sends all data newer than the serial in the Serial Query.

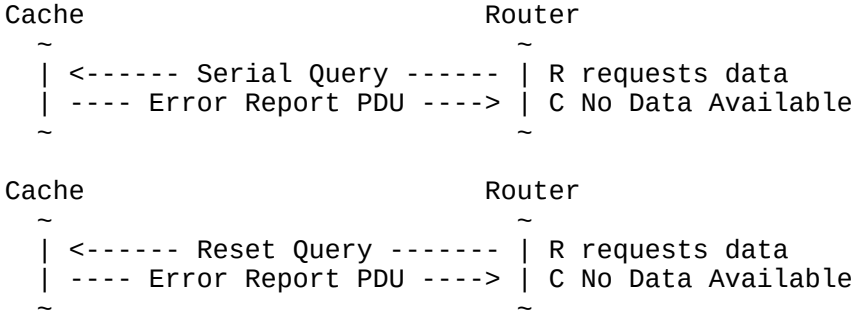
To limit the length of time a cache must keep old withdraws, a router MUST send either a Serial Query or a Reset Query periodically. See Section 6 for details on the required polling frequency.

8.3. No Incremental Update Available



The cache may respond to a Serial Query with a Cache Reset, informing the router that the cache cannot supply an incremental update from the Serial Number specified by the router. This might be because the cache has lost state, or because the router has waited too long between polls and the cache has cleaned up old data that it no longer believes it needed, or because the cache has run out of storage space and had to expire some old data early. Regardless of how this state arose, the cache replies with a Cache Reset to tell the router that it cannot honor the request. When a router receives this, the router SHOULD attempt to connect to any more-preferred caches in its cache list. If there are no more-preferred caches, it MUST issue a Reset Query and get an entire new load from the cache.

8.4. Cache Has No Data Available



The cache may respond to either a Serial Query or a Reset Query informing the router that the cache cannot supply any update at all. The most likely cause is that the cache has lost state, perhaps due to a restart, and has not yet recovered. While it is possible that a cache might go into such a state without dropping any of its active sessions, a router is more likely to see this behavior when it initially connects and issues a Reset Query while the cache is still [re]building its database.

When a router receives this kind of error, the router SHOULD attempt to connect to any other caches in its cache list, in preference order. If no other caches are available, the router MUST issue periodic Reset Queries until it gets a new usable load from the cache; maybe once a minute or less frequently so as not to DoS the cache.

9. Transport

The transport-layer session between a router and a cache carries the binary PDUs in a persistent reliable session.

To prevent cache spoofing and DoS attacks by illegitimate routers, it is highly desirable that the router and the cache be authenticated to each other. Integrity protection for payloads is also desirable to protect against monkey-in-the-middle (MITM) attacks. Unfortunately, there is no protocol to do so on all currently used platforms. Therefore, as of the writing of this document, there is no mandatory-to-implement transport which provides authentication and integrity protection.

To reduce exposure to dropped but non-terminated sessions, both caches and routers SHOULD enable keep-alives when available in the chosen transport protocol.

Should the cache or the router experience a transport stall (e.g. the peer advertised a TCP RCV.WND [RFC793] [RFC9293] of zero) for longer than three times the Retry Interval (a la BGP’s hold timer being three times the keepalive interval), an Error PDU 10, Transport Failure, should be sent and the transport session should be terminated.

A cache SHOULD NOT use a separate TCP segment for each PDU, but rather try to pack PDUs efficiently.

It is expected that, when the TCP Authentication Option (TCP-AO) [RFC5925] is available on all platforms deployed by operators, it will become the mandatory-to-implement transport.

Caches and routers MUST implement unprotected transport over TCP using a port, rpki-rtr (323); see Section 14. Operators SHOULD use procedural means, e.g., access control lists (ACLs), to reduce the exposure to authentication issues.

If unprotected TCP is the transport, the cache and routers MUST be on the same trusted and controlled network.

If available to the operator, caches and routers MUST use one of the following more protected protocols:

* Caches and routers SHOULD use TCP-AO transport [RFC5925] over the rpki-rtr port (323).

* Caches and routers MAY use Secure Shell version 2 (SSHv2) transport [RFC4252] using the normal SSH port (22). For an example, see Section 9.1.

* Caches and routers MAY use TCP MD5 transport [RFC2385] using the rpki-rtr port (323) if no other protected transport is available. Note that TCP MD5 has been obsoleted by TCP-AO [RFC5925].

* Caches and routers MAY use TCP over IPsec transport [RFC4301] using the rpki-rtr port (323).

* Caches and routers MAY use Transport Layer Security (TLS) transport [RFC8446] using port rpki-rtr-tls (324); see Section 14. Conformance to [BCP195] modern cipher suites is REQUIRED.

9.1. SSH Transport

To run over SSH, the client router first establishes an SSH transport connection using the SSHv2 transport protocol, and the client and server exchange keys for message integrity and encryption. The client then invokes the "ssh-userauth" service to authenticate the application, as described in the SSH authentication protocol [RFC4252]. Once the application has been successfully authenticated, the client invokes the "ssh-connection" service, also known as the SSH connection protocol.

After the ssh-connection service is established, the client opens a channel of type "session", which results in an SSH session.

Once the SSH session has been established, the application invokes the application transport as an SSH subsystem called "rpki-rtr". Subsystem support is a feature of SSHv2 and is not included in SSHv1. Running this protocol as an SSH subsystem avoids the need for the application to recognize shell prompts or skip over extraneous information, such as a system message that is sent at shell startup.

It is assumed that the router and cache have exchanged keys out of band by some reasonably secured means.

User authentication "publickey" MUST be supported; host authentication "hostbased" MAY be supported. Implementations MAY support password authentication "password". "None" authentication MUST NOT be used. Client routers SHOULD verify the public key of the cache to avoid MITM attacks.

9.2. TLS Transport

Client routers using TLS transport MUST present client-side certificates to authenticate themselves to the cache in order to allow the cache to manage their load by rejecting connections from unauthorized routers. In principle, any type of certificate and Certification Authority (CA) may be used; however, in general, cache operators will wish to create their own small-scale CA and issue certificates to each authorized router. This simplifies credential rollover; any unrevoked, unexpired certificate from the proper CA may be used.

Certificates used to authenticate client routers in this protocol MUST include a subjectAltName extension [RFC5280] containing one or more iPAddress identities; when authenticating the router’s certificate, the cache MUST check the IP address of the TLS connection against these iPAddress identities and SHOULD reject the connection if none of the iPAddress identities match the connection.

Routers MUST also verify the cache’s TLS server certificate, using subjectAltName dNSName identities as described in [RFC6125], to avoid MITM attacks. The rules and guidelines defined in [RFC6125] apply here, with the following considerations:

* Support for the DNS-ID identifier type (that is, the dNSName identity in the subjectAltName extension) is REQUIRED in rpki-rtr server and client implementations which use TLS. Certification authorities which issue rpki-rtr server certificates MUST support the DNS-ID identifier type, and the DNS-ID identifier type MUST be present in rpki-rtr server certificates.

* DNS names in rpki-rtr server certificates SHOULD NOT contain the wildcard character "*".

* rpki-rtr implementations which use TLS MUST NOT use Common Name (CN-ID) identifiers; a CN field may be present in the server certificate's subject name but MUST NOT be used for authentication within the rules described in [RFC6125].

* The client router MUST set its "reference identifier" (see Section 6.2 of [RFC6125]) to the DNS name of the rpki-rtr cache.

9.3. TCP MD5 Transport

If TCP MD5 is used, implementations MUST support key lengths of at least 80 printable ASCII octets, per Section 4.5 of [RFC2385]. Implementations MUST also support hexadecimal sequences of at least 32 characters, i.e., 128 bits.

Key rollover with TCP MD5 is problematic. Cache servers SHOULD support [RFC4808].

9.4. TCP-AO Transport

Implementations MUST support key lengths of at least 80 printable ASCII octets. Implementations MUST also support hexadecimal sequences of at least 32 characters, i.e., 128 bits. Message Authentication Code (MAC) lengths of at least 96 bits MUST be supported, per Section 5.1 of [RFC5925].

The cryptographic algorithms and associated parameters described in [RFC5926] MUST be supported.

10. Router-Cache Setup

A cache has the public authentication data for each router it is configured to support.

A router may be configured to peer with a selection of caches, and a cache may be configured to support a selection of routers. Each must have the name of, and authentication data for, each peer. In addition, in a router, this list has a non-unique preference value for each cache. This preference is intended to be based on proximity, a la RTT, not trust, preferred belief, et cetera. The client router attempts to establish a session with each potential serving cache in preference order and then starts to load data from the most preferred cache to which it can connect and authenticate. The router's list of caches has the following elements:

Preference: An unsigned integer denoting the router's preference to connect to that cache; the lower the value, the more preferred.

Name: The IP address or fully qualified domain name of the cache.

Cache Credential(s): Any credential (such as a public key) needed to authenticate the cache's identity to the router.

Router Credential(s): Any credential (such as a private key or certificate) needed to authenticate the router's identity to the cache.

Due to the distributed nature of the RPKI, caches simply cannot be rigorously synchronous. A client may hold data from multiple caches but MUST keep the data marked as to source, as later updates MUST affect the correct data.

Just as there may be more than one covering ROA from a single cache, there may be multiple covering ROAs from multiple caches. The results are as described in [RFC6811].

If data from multiple caches are held, implementations MUST NOT distinguish between data sources when performing validation of BGP announcements.

When a more-preferred cache becomes available, if resources allow, it would be prudent for the client to start fetching from that cache.

The router SHOULD attempt to maintain at least one set of data, regardless of whether it has chosen a different cache or established a new connection to the previous cache.

A client MAY drop the data from a particular cache when it is fully in sync with one or more other caches.

See Section 6 for details on what to do when the client is not able to refresh from a particular cache.

If a client loses connectivity to a cache it is using or otherwise decides to switch to a new cache, it SHOULD retain the data from the previous cache until it has a full set of data from one or more other caches. Note that this may already be true at the point of connection loss if the client has connections to more than one cache.

To keep load on Global RPKI services from unnecessary peaks, it is recommended that caches which fetch from the Global RPKI not do so all at the same times, e.g., on the hour. Choose a random time, perhaps the ISP's AS number modulo 60, and jitter the inter-fetch timing.

11. Races, Ordering, and Transactions

If a client applies individual PDUs as they are received from the cache, such that the PDUs are taken into account immediately for the purposes of BGP announcement validation, it is possible for a BGP announcement to be classified as invalid incorrectly. This is because subsequent PDUs received as part of the same cache response would lead to the BGP announcement being classified in some other way. See Section 11.1 for examples of when this might occur, Section 11.2 for a mandatory PDU ordering algorithm that acts to mitigate the effect of these race conditions on clients, and Section 11.3 for client implementation approaches to these problems.

11.1. Races

11.1.1. IPv4/IPv6 PDUs with Different Origins

If two BGP announcements exist for a given IP prefix destination, each with a different origin ASN, then two IP PDUs must be issued by a cache for those BGP announcements to be considered valid under route origin validation (ROV) [RFC6483]. If a client has only one of those PDUs, then the BGP announcement that is validated by the other PDU will be considered invalid until that other PDU is received.

11.1.2. IPv4/IPv6 PDUs for More-Specific Prefixes

For a given BGP announcement, there may be multiple IP PDUs that intersect the BGP announcement's prefix. If the client does not have all of these PDUs, it may classify the BGP announcement as invalid under ROV. For example, consider a scenario where there is a BGP announcement for 192.0.2.0/25 with an origin of AS64494, an IPv4 VRP for 192.0.2.0/24-24 with an origin of AS64495, and an IPv4 VRP for 192.0.2.0/25-25 with an origin of AS64494. A client that has received an announcement PDU for the first VRP only will classify the BGP announcement as invalid.

11.1.3. Withdrawal Before Announcement

If one of the underlying RPKI objects is modified in such a way that both a withdrawal PDU and an announcement PDU will be issued, and the withdrawal PDU is received by the client before the announcement PDU, this can lead to a BGP announcement being considered invalid until the announcement PDU is received. For example, consider a scenario where there is a BGP announcement for 192.0.2.0/25 with an origin of AS64494, an IPv4 VRP for 192.0.2.0/24-24 with an origin of AS64495, and an IPv4 VRP for 192.0.2.0/25-25 with an origin of AS64494. If a client receives a withdrawal PDU for the second VRP, followed by an announcement PDU that acts to replace the second VRP but with a greater max-length, then for the time between receipt of those two PDUs, the BGP announcement will be classified as invalid.

11.2. Ordering

For a protocol version 2 session, caches MUST use the ordering defined in this section when sending PDUs to clients as part of a cache response. Caches SHOULD use the ordering defined in this section for sessions using earlier protocol versions as well. This is because those versions did not prescribe a specific ordering for PDUs, and the benefits of ordering with respect to race conditions are the same for those earlier versions. An example scenario where an implementor would use another ordering for earlier protocol versions is where the implementation has an existing ordering it uses for those versions, and the implementor knows of clients that inadvertently depend on that ordering for some reason.

The ordering definitions in this section have been modelled on those from Section 4.3.3 of [RFC9582].

The ordering defined in this section is a total ordering. While a partial ordering could be defined that had the same benefits as far as race conditions are concerned, a total ordering makes it easier for clients to verify that the ordering is correct, and may also provide opportunities for client optimisations.

When using PDU values as integers for ordering comparisons, implementors should ensure that each value being used in a comparison is not inadvertently encoded using little-endian bit order, since such an encoding will lead to incorrect results.

PDUs with a lower integer PDU type precede PDUs with a higher integer PDU type. The sections that follow describe ordering as among PDUs of the same type.

11.2.1. IP PDUs

In order to semantically compare and sort IP PDUs, each PDU is mapped to an abstract data element comprising five integer values:

addr The first IP address of the IP prefix appearing in the PDU, as a 32-bit (IPv4) or 128-bit (IPv6) integer value.

plen The length of the IP prefix appearing in the PDU, as an integer value.

mlen The max length appearing in the PDU, as an integer value.

asn The origin ASN appearing in the PDU, as an integer value.

updt The integer 1, if the PDU is an announcement, and the integer 0, if the PDU is a withdrawal.

The equality or relative order of two IP PDUs can be tested by comparing their abstract representations.

The first order comparison is based on the updt value. Data elements with an updt value of 1 precede data elements with an updt value of 0. This addresses the problem described in Section 11.1.3.

The order of two IP PDUs with updt values of 1 is determined by the first non-equal comparison in the following list.

1. Data elements with a higher addr value precede data elements with a lower addr value.

2. Data elements with a higher mlen value precede data elements with a lower mlen value.

3. Data elements with a higher plen value precede data elements with a lower plen value.

4. Data elements with a higher asn value precede data elements with a lower asn value. This ensures that AS0 PDU announcements, which can be used to cause BGP announcements to become invalid under ROV, always appear after all other relevant PDU announcements.

The order of two IP PDUs with updt values of 0 is determined by the first non-equal comparison in the following list.

1. Data elements with a lower addr value precede data elements with a higher addr value.

2. Data elements with a lower mlen value precede data elements with a higher mlen value.

3. Data elements with a lower plen value precede data elements with a higher plen value.

4. Data elements with a lower asn value precede data elements with a higher asn value. This ensures that AS0 PDU withdrawals always appear before all other relevant PDU withdrawals.

For both announcement and withdrawal PDUs, the first three comparisons are included so as to address the problem described in Section 11.1.2.

11.2.2. Router Key PDUs

In order to semantically compare and sort Router Key PDUs, each PDU is mapped to an abstract data element comprising five values:

ski The Subject Key Identifier appearing in the PDU, as binary data (20 octets in length).

spkinfo The Subject Public Key Info appearing in the PDU, as DER-encoded data.

spkinfolen The number of octets in spkinfo.

asn The ASN appearing in the PDU, as an integer value.

updt The integer 1, if the PDU is an announcement, and the integer 0, if the PDU is a withdrawal.

The equality or relative order of two Router Key PDUs can be tested by comparing their abstract representations.

1. Data elements with an updt value of 1 precede data elements with an updt value of 0. This addresses the problem described in Section 11.1.3.

2. Data elements are then ordered based on their ski values. This ordering is lexicographical: each octet of binary data is treated as a symbol to compare, with the symbols ordered by their numerical value.

3. Data elements with a lower spkinfolen value precede data elements with a higher spkinfolen value.

4. Data elements are then ordered based on their spkinfo values. This ordering is the same as for the ski values.

5. Data elements with a lower asn value precede data elements with a higher asn value.

11.2.3. ASPA PDUs

In order to semantically compare and sort ASPA PDUs, each PDU is mapped to an abstract data element comprising two integer values:

casn The customer ASN appearing in the PDU, as an integer value.

updt The integer 1, if the PDU is an announcement, and the integer 0, if the PDU is a withdrawal.

The equality or relative order of two ASPA PDUs can be tested by comparing their abstract representations.

1. Data elements with an updt value of 1 precede data elements with an updt value of 0. This addresses the problem described in Section 11.1.3.

2. Data elements with a lower casn value precede data elements with a higher casn value.

11.3. Transactions

Clients are RECOMMENDED to apply PDUs only on receipt of an End of Data PDU. This implementation approach will ensure that the client is unaffected by the problems described in Section 11.1.

A client may be unable to apply PDUs only on receipt of an End of Data PDU, due e.g. to memory limitations. An alternative implementation approach that a client can use is to apply IP PDUs as a single group if their prefixes are equal. Since the mandatory ordering is such that those PDUs will appear in sequence, and such groups will generally be small, this should be easier for clients to support. A client that implements this approach is safe with respect to all of the problems described in Section 11.1.

A client that is unable to apply PDUs only on receipt of an End of Data PDU, and also unable to implement the alternative implementation approach described in this section, will be at risk of encountering the problems described in Section 11.1. While such problems will generally be transient, because a subsequent PDU received during the same synchronisation attempt should act to rectify the problem, there is a risk that those PDUs are not received due to an error during the

synchronisation attempt. In that scenario, clients SHOULD revert the application of any PDUs received during the failed synchronisation attempt.

11.4. Other Considerations

A client that applies PDUs transactionally (e.g. on receipt of an End of Data PDU) SHOULD consider limiting the period of time for which it will wait for the PDU that will close the transaction. This is especially important when initiating or resetting a session, since it may take some time to receive all relevant data from the cache. [todo: There were comments about adding guidance as to the timeout here, but how should that be determined? Can it be based on the expiry interval, or something like that?]

A client MAY verify the ordering of PDUs, and when an ordering error is encountered, send an Ordering Error PDU to the server and terminate the session. However, given that ordering issues lead only to transient problems, and also that resetting the session with the same cache is unlikely to fix the problem, clients should consider whether terminating the session is worthwhile. For example, if another cache is available, that may weigh in favour of terminating the current session and switching to that other cache.

Section 11.1 describes various scenarios where immediate application of PDUs can cause problems, depending on the order in which they are returned. A related problem is where an RPKI CA operator makes non-atomic changes to RPKI objects in such a way that the same ordering problem occurs, due to the cache receiving the updates at different times. However, RPKI CAs are able to make groups of changes take effect atomically, so in the common case where this problem arises, it will be due to limitations in CA implementations. Even in a case where a group of changes cannot be made to take effect atomically, it will be possible to order the changes at the CA such that no problems occur. For example, the operator could add a new ROA prior to removing an old ROA, allowing enough time between the operations that the risk of a cache or a client receiving both updates at the same time is low.

Related to the previous paragraph is the scenario where RPKI objects that need to be ordered in a specific way are issued by separate CAs. As with non-atomic operations by a single CA, CA operators in this situation must co-ordinate so as to ensure that the operations are ordered correctly and that there is sufficient time between the operations. CA operators in this situation should consider managing all such RPKI objects within a single CA, though, to avoid the problems that arise here.

A client that is generally unable to apply PDUs only on receipt of an End of Data PDU should consider whether it is possible to implement that behaviour at least in the context of new session initiation and session reset. Doing so will limit the risk of BGP announcements being used for a short period of time before being rejected due to the receipt of an invalidating PDU.

12. Error Codes

This section describes the meaning of the error codes. There is an IANA registry where valid error codes are listed; see [iana-err]. Errors which are considered fatal MUST cause the session to be dropped, and the router MUST flush all data learned from that cache.

0: Corrupt Data (fatal): The receiver believes the received PDU to be corrupt in a manner not specified by another error code.

1: Internal Error (fatal): The party reporting the error experienced some kind of internal error unrelated to protocol operation (ran out of memory, a coding assertion failed, et cetera).

2: No Data Available (non-fatal): The cache believes itself to be in good working order but is unable to answer either a Serial Query or a Reset Query because it has no useful data available at this time. This is likely to be a temporary error and most likely indicates that the cache has not yet completed pulling down an initial current data set from the Global RPKI system after some kind of event that invalidated whatever data it might have previously held (reboot, network partition, et cetera).

3: Invalid Request (fatal): The cache server believes the client’s request to be invalid.

4: Unsupported Protocol Version (non-fatal): The Protocol Version is not known by the receiver of the PDU. A session is not [re-]established, but data previously learned need not be flushed.

5: Unsupported PDU Type (fatal): The PDU Type is not known by the receiver of the PDU.

6: Withdrawal of Unknown Record (fatal): The received PDU has Flag=0, but a matching record ({Prefix, Len, Max-Len, AS} tuple for an IPvX PDU, or {SKI, AS, Subject Public Key} tuple for a Router Key PDU), or Customer Autonomous System for an ASPA PDU does not exist in the receiver’s database.

7: Duplicate Announcement Received (fatal): The received PDU has

Flag=1, but a matching record ({Prefix, Len, Max-Len, AS} tuple for an IPvX PDU, or {SKI, AS, Subject Public Key} tuple for a Router Key PDU), or Customer Autonomous System for an ASPA PDU is already active in the router.

8: Unexpected Protocol Version (fatal): The received PDU has a Protocol Version field that differs from the protocol version negotiated in Section 7.

9: ASPA Provider List Error (fatal): The received ASPA PDU has an incorrect list of Provider Autonomous System Numbers.

10: Transport Error (fatal): An error such as a stall (see Section 9) or other transport layer failure occurred.

11: Ordering Error (non-fatal): The received PDU does not conform with the ordering defined in Section 11.2.

13. Security Considerations

As this document describes a security protocol, many aspects of security interest are described in the relevant sections. This section points out issues which may not be obvious in other sections.

Cache Peer Identification: The router initiates a transport connection to a cache, which it identifies by either IP address or fully qualified domain name. Be aware that a DNS or address spoofing attack could make the correct cache unreachable. No session would be established, as the authorization keys would not match.

Cache Validation: In order for a collection of caches to provide a consistent view, they need to be given consistent trust anchors of the Certification Authorities to use in their internal validation process. Distribution of a consistent trust anchor set to validating caches is assumed to be out of band or specified elsewhere.

Transport Security: The RPKI relies on object, not server or transport, security. Trust anchor(s) are distributed to all caches through an out-of-band mechanism specified elsewhere. This can then be used by each cache to validate signed objects all the way up the tree. The inter-cache relationships are based on this object security model; hence, any inter-cache transport can be lightly protected.

However, this protocol assumes that the routers cannot do the

validation cryptography. Hence, the last link, from cache to router, SHOULD be secured by server authentication and transport-level security to prevent monkey in the middle attacks; though it might not be. Not using transport security is dangerous, as server authentication and transport have very different threat models than object security.

So the strength of the trust relationship and the transport between the router(s) and the cache(s) are critical. You're betting your routing on this.

While we cannot say the cache must be on the same LAN, if only due to the issue of an enterprise wanting to offload the cache task to their upstream ISP(s), locality, trust, and control are very critical issues here. The cache(s) really SHOULD be as close, in the sense of controlled and protected (against DDoS, MITM) transport, to the router(s) as possible. It also SHOULD be topologically close so that a minimum of validated routing data are needed to bootstrap a router's access to a cache.

Authenticating transport protocols (i.e. not raw TCP) will authenticate the identity of the cache server to the router client, and vice versa, before any data are exchanged.

Transports which cannot provide the necessary authentication and integrity (see Section 9) must rely on network design and operational controls to provide protection against spoofing/corruption attacks. As pointed out in Section 9, TCP-AO is the long-term plan. Protocols which provide integrity and authenticity SHOULD be used, and if they cannot, i.e., TCP is used as the transport, the router and cache MUST be on the same trusted, controlled network.

14. IANA Considerations

This section only discusses updates required in the existing IANA protocol registries to accommodate version 2 of this protocol. See [RFC8210] for IANA considerations of the previous (version 1) protocol.

All of the PDU types in the IANA "rpki-rtr-pdu" registry [iana-pdu] in protocol versions 0 and 1 are also allowed in protocol version 2, with the addition of the new ASPA PDU.

The "rpki-rtr-error: registry [iana-err] should be updated as follows:

Error Code	Description
0	Corrupt Data
1	Internal Error
2	No Data Available
3	Invalid Request
4	Unsupported Protocol Version
5	Unsupported PDU Type
6	Withdrawal of Unknown Record
7	Duplicate Announcement Received
8	Unexpected Protocol Version
9	ASPA Provider List Error
10	Transport Failure
11	Ordering Error
11-254	Unassigned
255	Reserved

The "rpki-rtr-pdu" registry [iana-pdu] has been updated as follows:

Protocol Version	PDU Type	Description
0-2	0	Serial Notify
0-2	1	Serial Query
0-2	2	Reset Query
0-2	3	Cache Response
0-2	4	IPv4 Prefix
0-2	6	IPv6 Prefix
0-2	7	End of Data
0-2	8	Cache Reset
0	9	Reserved
1-2	9	Router Key
0-2	10	Error Report
0-1	11	Reserved
2	11	ASPA
0-2	255	Reserved

15. 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 [RFC7942]. 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 [RFC7942], "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".

15.1. rpki-rtr-demo

* Responsible Organization: APNIC
* Location: https://github.com/APNIC-net/rpki-rtr-demo
* Description: This implementation supports the behaviour described in this document.
* Level of Maturity: This is a proof-of-concept implementation.
* Coverage: This implementation includes all of the features described in this specification, except for TCP-AO.
* Contact Information: Tom Harrison, tomh@apnic.net

16. References

16.1. Normative References

[BCP195] Best Current Practice 195, <https://www.rfc-editor.org/info/bcp195>. At the time of writing, this BCP comprises the following:

Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <https://www.rfc-editor.org/info/rfc8996>.

Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

[I-D.ietf-sidrops-aspa-profile]
Azimov, A., Uskov, E., Bush, R., Snijders, J., Housley, R., and B. Maddison, "A Profile for Autonomous System Provider Authorization", Work in Progress, Internet-Draft, draft-ietf-sidrops-aspa-profile-20, 18 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-profile-20>.

[iana-err] IANA, "rpki-rtr-error", <https://www.iana.org/assignments/rpki#rpki-rtr-error>.

[iana-pdu] IANA, "rpki-rtr-pdu", <https://www.iana.org/assignments/rpki#rpki-rtr-pdu>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[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>.

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, DOI 10.17487/RFC2385, August 1998, <https://www.rfc-editor.org/info/rfc2385>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/info/rfc5925>.

[RFC5926] Lebovitz, G. and E. Rescorla, "Cryptographic Algorithms for the TCP Authentication Option (TCP-AO)", RFC 5926, DOI 10.17487/RFC5926, June 2010, <https://www.rfc-editor.org/info/rfc5926>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <https://www.rfc-editor.org/info/rfc6125>.

[RFC6483] Huston, G. and G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)", RFC 6483, DOI 10.17487/RFC6483, February 2012, <https://www.rfc-editor.org/info/rfc6483>.

[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>.

[RFC6810] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol", RFC 6810, DOI 10.17487/RFC6810, January 2013, <https://www.rfc-editor.org/info/rfc6810>.

[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>.

[RFC8210] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1", RFC 8210, DOI 10.17487/RFC8210, September 2017, <https://www.rfc-editor.org/info/rfc8210>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8608] Turner, S. and O. Borchert, "BGPsec Algorithms, Key Formats, and Signature Formats", RFC 8608, DOI 10.17487/RFC8608, June 2019, <https://www.rfc-editor.org/info/rfc8608>.

[RFC8635] Bush, R., Turner, S., and K. Patel, "Router Keying for BGPsec", RFC 8635, DOI 10.17487/RFC8635, August 2019, <https://www.rfc-editor.org/info/rfc8635>.

16.2. Informative References

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <https://www.rfc-editor.org/info/rfc1996>.

[RFC4808] Bellovin, S., "Key Change Strategies for TCP-MD5", RFC 4808, DOI 10.17487/RFC4808, March 2007, <https://www.rfc-editor.org/info/rfc4808>.

[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>.

[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>.

[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>.

[RFC793] Postel, J., "Transmission Control Protocol", RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8654] Bush, R., Patel, K., and D. Ward, "Extended Message Support for BGP", RFC 8654, DOI 10.17487/RFC8654, October 2019, <https://www.rfc-editor.org/info/rfc8654>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[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>.

Acknowledgements

The authors wish to thank Nils Bars, Steve Bellovin, Oliver Borchert, Mohamed Boucadair, Tim Bruijnzeels, Ralph Covelli, Roman Danyliw, Rex Fernando, Richard Hansen, Martin Hoffmann, Paul Hoffman, Fabian Holler, Russ Housley, Claudio Jeker, Pradosh Mohapatra, Keyur Patel, David Mandelberg, Sandy Murphy, Robert Raszuk, Andreas Reuter, Thomas Schmidt, John Scudder, Job Snijders, Ruediger Volk, Matthias Waehlisch, and David Ward. Particular thanks go to Hannes Gredler for showing us the dangers of unnecessary fields.

No doubt this list is incomplete. We apologize to any contributor whose name we missed.

Authors' Addresses

Randy Bush
Arrcus, DRL, & IIJ Research
Email: randy@psg.com

Rob Austein
Dragon Research Labs
Email: sra@hactrn.net