Network Working Group R. Bush
Internet-Draft IIJ
Intended status: Standards Track R. Austein
Expires: February 16, 2011 ISC
August 15, 2010

The RPKI/Router Protocol
draft-ietf-sidr-rpki-rtr-01

Abstract

In order to formally validate the origin ASes of BGP announcements, routers need a simple but reliable mechanism to receive RPKI [I-D.ietf-sidr-arch] or analogous prefix origin data from a trusted cache. This document describes a protocol to deliver validated prefix origin data to routers over ssh.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction 3
2. Deployment Structure 3
3. Operational Overview 3
4. Protocol Data Units (PDUs) 4
4.1. Serial Notify 5
4.2. Serial Query 5
4.3. Reset Query 6
4.4. Cache Response 6
4.5. IPv4 Prefix 7
4.6. IPv6 Prefix 8
4.7. End of Data 8
4.8. Cache Reset 9
4.9. Error Report 9
4.10. Fields of a PDU 10
5. Protocol Sequences 11
5.1. Start or Restart 11
5.2. Typical Exchange 12
5.3. No Incremental Update Available 13
5.4. Cache has No Data Available 13
6. SSH Transport 14
7. Router-Cache Set-Up 14
8. Deployment Scenarios 16
9. Error Codes 16
10. Security Considerations 17
11. Glossary 18
12. IANA Considerations 18
13. Acknowledgments 19
14. References 19
14.1. Normative References 19
14.2. Informative References 19
Authors' Addresses 20

1. Introduction

In order to formally validate the origin ASes of BGP announcements, routers need a simple but reliable mechanism to receive RPKI [I-D.ietf-sidr-arch] or analogous formally validated prefix origin data from a trusted cache. This document describes a protocol to deliver validated prefix origin data to routers over ssh.

Section 2 describes the deployment structure and Section 3 then presents an operational overview. The binary payloads of the protocol are formally described in Section 4, and the expected PDU sequences are described in Section 5. The transport protocol is described in Section 6. Section 7 details how routers and caches are configured to connect and authenticate. Section 8 describes likely deployment scenarios. The traditional security and IANA considerations end the document.

The protocol is extensible to support new PDUs with new semantics when and as needed, as indicated by deployment experience. PDUs are versioned should deployment experience call for change.

2. 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, RPKI publication repositories, e.g. the IANA, RIRs, NIRs, and ISPs, see [I-D.ietf-sidr-repos-struct].

Local Caches: A local set of one or more collected and verified non-authoritative caches. A relying party, e.g. router or other client, MUST have a formally authenticated trust relationship with, and a secure transport channel to, any non-authoritative 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 are mechanisms for the router to assure itself of the authenticity of the cache and to authenticate itself to the cache.

3. Operational Overview

A router establishes and keeps open an authenticated connection to a cache with which it has a client/server relationship. It is

configured with a semi-ordered list of caches, and establishes a connection to the most preferred cache, or set of caches, which accepts one.

Periodically, the router sends to the cache the serial number of the highest numbered data record it has received from that cache, i.e. the router's current serial number. When a router establishes a new connection to a cache, or wishes to reset a current relationship, it sends a Reset Query.

The Cache responds with all data records which have serial numbers greater than that in the router's query. This may be the null set, in which case the End of Data PDU is still sent. Note that 'greater' must take wrap-around into account, see [RFC1982].

When the router has received all data records from the cache, it sets its current serial number to that of the serial number in the End of Data PDU.

When the cache updates its database, it sends a Notify message 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 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 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.

4. 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 5.

4.1. Serial Notify

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

```
   0          8          16         24        31
   .-------------------------------------------.
   | Protocol |   PDU    |                     |
   | Version  |   Type   |   reserved = zero   |
   |    0     |    0     |                     |
   +-------------------------------------------+
   |                                           |
   |                Length=12                  |
   |                                           |
   +-------------------------------------------+
   |                                           |
   |               Serial Number               |
   |                                           |
   `-------------------------------------------'
```

4.2. Serial Query

Serial Query: The router sends Serial Query to ask the cache for all payload PDUs which have serial numbers higher than the serial number in the Serial Query.

The cache replies to this query with a Cache Response PDU (Section 4.4) if the cache has a record of the changes since the serial number specified by the router. If there have been no changes since the router last queried, the cache responds with an End Of Data PDU. 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 4.8).

```
   0          8          16         24        31
   .-------------------------------------------.
   | Protocol |   PDU    |                     |
   | Version  |   Type   |   reserved = zero   |
   |    0     |    1     |                     |
   +-------------------------------------------+
   |                                           |
   |                 Length=12                 |
   |                                           |
   +-------------------------------------------+
   |                                           |
   |               Serial Number               |
   |                                           |
   `-------------------------------------------'
```

4.3. Reset Query

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 4.4).

```
   0          8          16         24        31
   .-------------------------------------------.
   | Protocol |   PDU    |                     |
   | Version  |   Type   |    reserved = zero  |
   |    0     |    2     |                     |
   +-------------------------------------------+
   |                                           |
   |                 Length=8                  |
   |                                           |
   `-------------------------------------------'
```

4.4. Cache Response

Cache Response: The cache responds with zero or more payload PDUs. When replying to a Serial Query request (Section 4.2), the cache sends the set of all data records it has with serial numbers greater than that sent by the client router. When replying to a Reset Query, the cache sends the set of all data records it has; in this case the withdraw/announce field in the payload PDUs MUST have the value 1 (announce).

```
   0          8          16         24        31
   .-------------------------------------------.
   | Protocol |   PDU    |                     |
   | Version  |   Type   |   reserved = zero   |
   |    0     |    3     |                     |
   +-------------------------------------------+
   |                                           |
   |                 Length=8                  |
   |                                           |
   `-------------------------------------------'
```

4.5. IPv4 Prefix

```
0          8          16         24        31
.-------------------------------------------.
| Protocol |   PDU    |                     |
| Version  |   Type   |    reserved = zero  |
|    0     |    4     |                     |
+-------------------------------------------+
|                                           |
|                 Length=20                 |
|                                           |
+-------------------------------------------+
|          |  Prefix  |   Max    |          |
|  Flags   |  Length  |  Length  |   zero   |
|          |   0..32  |   0..32  |          |
+-------------------------------------------+
|                                           |
|                IPv4 prefix                |
|                                           |
+-------------------------------------------+
|                                           |
|         Autonomous System Number          |
|                                           |
`-------------------------------------------'
```

Due to the nature of the RPKI and the IRR, there can be multiple identical IPvX PDUs. A router MUST be prepared to receive multiple identical record announcements and MUST NOT consider a record to have been deleted until it has received a corresponding number of withdrawals or a reset is performed Hence the router will likely keep an internal reference count on each IPvX PDU.

In the RPKI, nothing prevents a signing certificate from issuing two identical ROAs, and nothing prohibits the existence of two identical route: or route6: objects in the IRR. In this case there would be no semantic difference between the objects, merely a process redundancy.

In the RPKI, there is also an actual need for what will appear to the router as identical IPvX PDUs. This occurs when an upstream certificate is being reissued or a site is changing providers, often a 'make and break' situation. The ROA is identical in the router sense, i.e. has the same {prefix, len, max-len, asn}, but has a different validation path in the RPKI. This is important to the RPKI, but not to the router.

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

4.6. IPv6 Prefix

```
   0          8          16         24        31
   .-------------------------------------------.
   | Protocol |   PDU    |                     |
   | Version  |   Type   |    reserved = zero  |
   |    0     |    6     |                     |
   +-------------------------------------------+
   |                                           |
   |                 Length=32                 |
   |                                           |
   +-------------------------------------------+
   |          |  Prefix  |   Max   |          |
   |  Flags   |  Length  |  Length |   zero   |
   |          |   0..32  |  0..128 |          |
   +-------------------------------------------+
   |                                           |
   +---                                     ---+
   |                                           |
   +---            IPv6 prefix              ---+
   |                                           |
   +---                                     ---+
   |                                           |
   +-------------------------------------------+
   |                                           |
   |         Autonomous System Number          |
   |                                           |
   `-------------------------------------------'
```

4.7. End of Data

End of Data: Cache tells router it has no more data for the request.

```
   0          8          16         24        31
   .-------------------------------------------.
   | Protocol |   PDU    |                     |
   | Version  |   Type   |   reserved = zero   |
   |    0     |    7     |                     |
   +-------------------------------------------+
   |                                           |
   |                 Length=12                 |
   |                                           |
   +-------------------------------------------+
   |                                           |
   |               Serial Number               |
   |                                           |
   `-------------------------------------------'
```

4.8. Cache Reset

The cache may respond to a Serial Query informing 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 switch to a different cache.

```
0          8          16         24        31
.-------------------------------------------.
| Protocol |   PDU     |                     |
| Version  |   Type    |    reserved = zero  |
|    0     |    8      |                     |
+-------------------------------------------+
|                                           |
|                 Length=8                  |
|                                           |
`-------------------------------------------'
```

4.9. Error Report

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

The Error Number is described in Section 9.

If the error is not associated with any particular PDU, the Erroneous PDU field should be empty and the Length of Encapsulated PDU field should be zero.

The diagnostic text is optional, if not present the Length of Error Text field should be zero. If error text is present, it SHOULD be a string in US-ASCII, for maximum portability; if non-US-ASCII characters are absolutely required, the error text MUST use UTF-8 encoding.

```
   0          8          16         24        31
   .-------------------------------------------.
   | Protocol |   PDU    |                     |
   | Version  |   Type   |    Error Number     |
   |    0     |    10    |                     |
   +-------------------------------------------+
   |                                           |
   |                  Length                   |
   |                                           |
   +-------------------------------------------+
   |                                           |
   |       Length of Encapsulated PDU          |
   |                                           |
   +-------------------------------------------+
   |                                           |
   ~           Copy of Erroneous PDU           ~
   |                                           |
   +-------------------------------------------+
   |                                           |
   |           Length of Error Text            |
   |                                           |
   +-------------------------------------------+
   |                                           |
   |              Arbitrary Text               |
   |                    of                     |
   ~          Error Diagnostic Message         ~
   |                                           |
   `-------------------------------------------'
```

4.10. Fields of a PDU

PDUs contain the following data elements:

Protocol Version: An ordinal, currently 0, denoting the version of this protocol.

Serial Number: The serial number of the RPKI Cache when this ROA was received from the cache's up-stream cache server or gathered from the global RPKI. A cache increments its serial number when completing an rcynic from a parent cache. See [RFC1982] on DNS Serial Number Arithmetic for too much detail on serial number arithmetic.

Length: A 32 bit ordinal which has as its value the count of the bytes in the entire PDU, including the eight bytes of header which end with the length field.

Flags: The lowest order bit of the Flags field is 1 for an announcement and 0 for a withdrawal, whether this PDU announces a new right to announce the prefix or withdraws a previously announced right. A withdraw effectively deletes one previously announced IPvX Prefix PDU with the exact same Prefix, Length, Max-Len, and ASN.

Prefix Length: An ordinal denoting the shortest prefix allowed for the prefix.

Max Length: An ordinal denoting the longest prefix allowed by the prefix. This MUST NOT be less than the Prefix Length element.

Prefix: The IPv4 or IPv6 prefix of the ROA.

Autonomous System Number: ASN allowed to announce this prefix, a 32 bit ordinal.

Zero: Fields shown as zero or reserved MUST be zero. The value of such a field MUST be ignored on receipt.

5. Protocol Sequences

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

5.1. Start or Restart

```
   Cache                         Router
     ~                             ~
     | <----- Reset Query -------- | R requests data
     |                             |
     | ----- Cache Response -----> | C confirms request
     | ------- IPvX Prefix ------> | C sends zero or more
     | ------- IPvX Prefix ------> |   IPv4 and IPv6 Prefix
     | ------- IPvX Prefix ------> |   Payload PDUs
     | ------  End of Data ------> | C sends End of Data
     |                             |   and sends new serial
     ~                             ~
```

When a transport session is first established, the router sends a Reset Query and the cache responds with a data sequence of all data it contains.

This 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.

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 no less frequently than once an hour. This also acts as a keep alive at the application layer.

5.2. Typical Exchange

```
   Cache                         Router
     ~                             ~
     | -------- Notify ----------> |  (optional)
     |                             |
     | <----- Serial Query ------- | R requests data
     |                             |
     | ----- Cache Response -----> | C confirms request
     | ------- IPvX Prefix ------> | C sends zero or more
     | ------- IPvX Prefix ------> |   IPv4 and IPv6 Prefix
     | ------- IPvX Prefix ------> |   Payload PDUs
     | ------  End of Data ------> | C sends End of Data
     |                             |   and sends new serial
     ~                             ~
```

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 SHOULD 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, 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 no less frequently than once an hour.

5.3. No Incremental Update Available

```
   Cache                         Router
     ~                             ~
     | <-----  Serial Query ------ | R requests data
     | ------- Cache Reset ------> | C cannot supply update
     |                             |   from specified serial
     | <------ Reset Query ------- | R requests new data
     | ----- Cache Response -----> | C confirms request
     | ------- IPvX Prefix ------> | C sends zero or more
     | ------- IPvX Prefix ------> |   IPv4 and IPv6 Prefix
     | ------- IPvX Prefix ------> |   Payload PDUs
     | ------  End of Data ------> | C sends End of Data
     |                             |   and sends new serial
     ~                             ~
```

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 needs, 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.

5.4. Cache has No Data Available

```
   Cache                         Router
     ~                             ~
     | <-----  Serial Query ------ | R requests data
     | ---- Error Report PDU ----> | C cannot supply update
     ~                             ~
```

```
   Cache                         Router
     ~                             ~
     | <-----  Reset Query ------- | R requests data
     | ---- Error Report PDU ----> | C cannot supply update
     ~                             ~
```

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 rebuilding 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.

6. SSH Transport

The transport layer session between a router and a cache carries the binary Protocol Data Units (PDUs) in a persistent SSH session.

To run over SSH, the client router first establishes an SSH transport connection using the SSH 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 RFC 4252 [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 SSH version 2 (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 start-up.

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

7. Router-Cache Set-Up

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 server in order of preference. This preference merely denotes proximity, not trust, preferred belief, etc. 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 ordinal 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.

Key: The public ssh key of the cache.

MyKey: The private ssh key of this client.

Due to the distributed nature of the RPKI, caches simply can not 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 the same as described in [I-D.ietf-sidr-roa-validation] as follows:

- Data from no cache has a covering ROA, so the result is 'unknown'.

- One or more closest covering ROAs exist but none match the AS of the BGP update, so the result is 'invalid'.

- One or more closely covering ROAs have an AS match, while other equally closely covering ROAs do not have an AS match, so the result is 'valid'.

- There are multiple covering ROAs but none with an AS that matches the BGP annoucement, so the result is 'invalid'.

When a more preferred cache becomes available, if resources allow, it would be prudent for the client to start fetching from that cache. The client MAY drop the data from older, less preferred cache(s) when it is fully in synch with one or more other caches.

When a client loses connectivity to the cache it is currently 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 some other cache. It should do this regardless of whether it has

chosen a different cache or established a new connection to the previous cache. However, a configurable timer MUST be provided to bound how long it will retain the "stale" data.

8. Deployment Scenarios

For illustration, we present three likely deployment scenarios.

Small End Site: The small multi-homed end site may wish to outsource the RPKI cache to one or more of their upstream ISPs. They would exchange authentication material with the ISP using some out of band mechanism, and their router(s) would connect to one or more up-streams' caches. The ISPs would likely deploy caches intended for customer use separately from the caches with which their own BGP speakers peer.

Large End Site: A larger multi-homed end site might run one or more caches, arranging them in a hierarchy of client caches, each fetching from a serving cache which is closer to the global RPKI. They might configure fall-back peerings to up-stream ISP caches.

ISP Backbone: A large ISP would likely have one or more redundant caches in each major PoP, and these caches would fetch from each other in an ISP-dependent topology so as not to place undue load on the global RPKI publication infrastructure.

Experience with large DNS cache deployments has shown that complex topologies are ill-advised as it is easy to make errors in the graph, e.g. not maintaining a loop-free condition.

Of course, these are illustrations and there are other possible deployment strategies. It is expected that minimizing load on the global RPKI servers will be a major consideration.

To keep load on global RPKI services from unnecessary peaks, it is recommended that primary caches which load from the distributed 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.

9. Error Codes

This section contains a preliminary list of error codes. The authors expect additions to this section during development of the initial implementations. Eventually, these error codes will probably need to reside in an IANA registry.

0: Reserved.

1: Internal Error: 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: 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, etcetera).

10. 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 Validation: In order for a collection of caches as described in Section 8 to guarantee a consistent view, they need to be given consistent trust anchors to use in their internal validation process. Distribution of a consistent trust anchor is assumed to be out of band.

Cache Peer Identification: The router initiates an ssh transport session 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.

Transport Security: The RPKI relies on object, not server or transport, trust. I.e. the IANA root trust anchor is distributed to all caches through some out of band means, and can then be used by each cache to validate certificates and ROAs all the way down the tree. The inter-cache relationships are based on this object security model, hence the inter-cache transport can be lightly protected.

But this protocol document assumes that the routers can not do the validation cryptography. Hence the last link, from cache to router, is secured by server authentication and transport level security. This 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 can not say the cache must be on the same LAN, if only due to the issue of an enterprise wanting to off-load 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.

11. 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, RIRs, NIRs, and ISPs, see [I-D.ietf-sidr-repos-struct].

Non-authorative Cache: A coalesced copy of the RPKI which is periodically fetched/refreshed directly or indirectly from the global RPKI using the [RFC5781] protocol/tools

Cache: The rcynic system is used to gather the distributed data of the RPKI into a validated cache. Trusting this cache further is a matter between the provider of the cache and a relying party.

Serial Number: A 32-bit monotonically increasing ordinal which wraps from 2^32-1 to 0. It denotes the logical version of a cache. A cache increments the value by one when it successfully updates its data from a parent cache or from primary RPKI data. As a cache is rcynicing, new incoming data, and implicit deletes, are marked with the new serial but MUST not be sent until the fetch is complete. A serial number is not commensurate between caches, nor need it be maintained across resets of the cache server. See [RFC1982] on DNS Serial Number Arithmetic for too much detail on serial number arithmetic.

12. IANA Considerations

This document requests the IANA to create a registry for PDU types.

This document requests the IANA to create a registry for Error Codes.

In addition, a registry for Version Numbers would be needed if new Version Number is defined in a new RFC.

Note to RFC Editor: this section may be replaced on publication as an RFC.

13. Acknowledgments

The authors wish to thank Steve Bellovin, Rex Fernando, Russ Housley, Pradosh Mohapatra, Keyur Patel, Sandy Murphy, Robert Raszuk, John Scudder, Ruediger Volk, and David Ward. Particular thanks go to Hannes Gredler for showing us the dangers of unnecessary fields.

14. References

14.1. Normative References

[I-D.ietf-sidr-roa-validation]
Huston, G. and G. Michaelson, "Validation of Route Origination using the Resource Certificate PKI and ROAs", draft-ietf-sidr-roa-validation-06 (work in progress), May 2010.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4252] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.

[RFC5781] Weiler, S., Ward, D., and R. Housley, "The rsync URI Scheme", RFC 5781, February 2010.

14.2. Informative References

[I-D.ietf-sidr-arch]
Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", draft-ietf-sidr-arch-09 (work in progress), October 2009.

[I-D.ietf-sidr-repos-struct]
Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", draft-ietf-sidr-repos-struct-04 (work in progress),

May 2010.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.

Authors' Addresses

Randy Bush
Internet Initiative Japan, Inc.
5147 Crystal Springs
Bainbridge Island, Washington 98110
US

Phone: +1 206 780 0431 x1
Email: randy@psg.com

Rob Austein
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
USA

Email: sra@isc.org