XML Schemas for Reverse DNS Management

Abstract

This document defines an Extensible Markup Language (XML) schema for reverse DNS management in a tightly controlled Representational State Transfer (REST) environment. This document describes a schema that has been developed and deployed by ICANN in a "RESTful" system since 2011 and is being used by the registries responsible for reverse DNS (rDNS) delegations underneath IN-ADDR.ARPA and IP6.ARPA through an HTTPS transaction that is mediated by an X.509 certificate.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7745.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
1. Introduction

This document defines an Extensible Markup Language (XML) schema for reverse DNS management in a tightly controlled Representational State Transfer (REST) [REST] environment. This document describes a schema that has been developed and deployed by ICANN in a "RESTful" system since 2011 and is being used by the registries responsible for reverse DNS (rDNS) delegations underneath IN-ADDR.ARPA [RFC1034] and IP6.ARPA [RFC3596] through an HTTPS [RFC2818] transaction that is mediated by an X.509 [RFC5280] certificate.

As DNSSEC [RFC4033] adoption progresses, the necessity to interact with a delegation in the IN-ADDR.ARPA and IP6.ARPA zones becomes more frequent given that updates to DS records in the parent zone for child delegations follow the key rollover and expiry of the child zone. The modification of such critical areas at a relative high frequency requires a system that allows the administrative holders of such delegations to make such changes in a secure and trustworthy manner where the chain of trust for submitting the necessary information remains unbroken between the IN-ADDR.ARPA and IP6.ARPA zone maintainers and the zone customers.

At the request of the Regional Internet Registries (RIRs) to automate reverse DNS updates with ICANN, a REST-based HTTPS service was deployed that:

- Provides for a secure, authenticated mechanism to update zone data (NS and DS records)
- Provides a well-formed data structure for both the IN-ADDR.ARPA and IP6.ARPA zones
- Allows for "out-of-band" acknowledgement and notification of updates
2. 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].

3. Implementation

The implemented system allows the entity responsible for its rDNS
delegations to effect changes in the reverse DNS zones IN-ADDR.ARPA
and IP6.ARPA by submitting an XML document to an atomic RESTful
service via an HTTPS [RFC2818] connection. In this service, the
HTTPS layer provides the end-to-end security of the transaction, and
it further provides authentication by use of mandatory X.509
[RFC5280] client certificates with a known server certificate issued
by a Certification Authority administered by the service operator.

Certificates for use in this system, issued by the system operator,
are specific to the entity responsible for the delegations in the
zone.

Updates are made to the system by using the HTTP GET, PUT, and DELETE
operations over HTTP 1.1 [RFC7231] via HTTPS [RFC2818] only. These
operations are sent to a resource Uniform Resource Identifier (URI)
in the form of:

    https://host.example.org/<ipversion>/<zone>

A synthetic example of an XML document submitted to the deployed
system might take the following form (including all optional
attributes) as per the schema in Appendix A.
<zone xmlns="http://download.research.icann.org/rdns/1.1"
    name="10.in-addr.arpa" cust="IANA" ipversion="ipv4"
    version="1.1" modified="2012-01-18T01:00:06"
    state="active" href="https://host.example.org/ipv4/10">
  <nserver>
    <fqdn>BLACKHOLE-1.IANA.ORG.</fqdn>
  </nserver>
  <nserver>
    <fqdn>BLACKHOLE-2.IANA.ORG.</fqdn>
  </nserver>
  <ds>
    <rdata>33682 5 1 ea8afbf5fce7caf381ab101039</rdata>
  </ds>
  <ds>
    <rdata>33682 5 2 7d44874f1d93aaceb793a88001739a</rdata>
  </ds>
</zone>

When PUT and DELETE operations are used, the well-formed XML is required to be sent with the appropriate content-length headers. The GET operation requires only the URI.

One requirement of the system was to allow the separation of update and approval with an out-of-band notification mechanism. When such options are configured for a customer of the service, submitted updates may be queued for later approval. When a customer has queued updates pending approval, the customer may submit a GET request to retrieve either an individual entry or a full listing of all queued entries.

To fetch a listing of the customer’s queue, the customer would GET a URI in the form of:

https://host.example.org/queuelist

To fetch an individual queue entry, the customer would GET the canonical URL (as per the schema) for this queue record:

https://host.example.org/queue/<identifier>

Where <identifier> is a system-generated and system-specific value that identifies this particular queue entry. All XML returned from queue-based operations ('queue' and 'queuelist') would return an XML document following the specification in Appendix B. A synthetic example from a GET of 'queuelist' would be:
<queuelist xmlns="http://download.research.icann.org/rq/1.0" version="1.0">
<queue xmlns="http://download.research.icann.org/rq/1.0" name="10.in-addr.arpa" cust="IANA" ipversion="ipv4" version="1.0" submitted="2013-01-11T05:22:15" state="pending" method="PUT" ack="https://host.example.org/ack/25a531f50e5ba45" href="https://host.example.org/queue/25a531f50e5ba45">
<nserver>
  <fqdn>BLACKHOLE-1.IANA.ORG.</fqdn>
</nserver>
<nserver>
  <fqdn>BLACKHOLE-2.IANA.ORG.</fqdn>
</nserver>
<ds>
  <rdata>33682 5 1 ea8af5fce7caf381ab101039</rdata>
</ds>
<ds>
  <rdata>33682 5 2 7d44874f1d93aaceb793a88001739a</rdata>
</ds>
</queue>
</queuelist>

4. Security Considerations

This document provides an XML schema for facilitating the management of reverse DNS delegations in the IN-ADDR.ARPA and IP6.APRA zones. The schema itself contains no authentication data, and all other information contained is considered public data as it is either published in DNS or propagated to other public information sources like WHOIS.

The system that implements this XML schema requires HTTPS to be used and also uses known server and client X.509 certificates for authentication to protect against message modification, message insertion/deletion, man-in-the-middle, and replay attacks. Any DoS-type attack vectors and the authorisation of which delegations the X.509 certificate authentication sessions can affect are out of scope for this document.

5. References

5.1. Normative References

5.2. Informative References


Appendix A. Schema Definition for rDNS Updates

The following Schema, used for PUT, GET, and DELETE operations, is an XML document using the RelaxNG Compact [RELAXNG] specification.

default namespace = "http://download.research.icann.org/rdns/1.1"

# A document may either be a single zone (update) or a collection of zones (view)
start = zone | zonelist | zonereflist

# A list of zone names for view only.
zonereflist = element zonereflist {
    attribute version {
        xsd:decimal { minInclusive="1.1" fractionDigits="1" }
    },
    zoneref*
}

# A bulk list of zones for view only.
zonelist = element zonelist {
    attribute version {
        xsd:decimal { minInclusive="1.1" fractionDigits="1" }
    },
    zone*
}

# A zone reference (accepted by REST engine for query)
zoneref = element zoneref {
    attribute name { text },
    attribute href { xsd:anyURI }
}

# A single zone record
zone = element zone {
    # The zone record’s name, e.g., 10.in-addr.arpa
    attribute name { text },
    # The customer (optional); derived from known state.
    attribute cust { text }?,
    # The canonical URL for this zone record (optional)
    attribute href { xsd:anyURI }?,
    # The IP version of the address for the zone record (optional)
    attribute ipversion { "ipv4" | "ipv6" }?,
    # The administrative state of the zone (optional)
    attribute state { "active" | "pending" | "error" }?,
    # The last modified timestamp in UTC (optional)
    attribute modified { xsd:dateTime }?,
    # The schema version (optional)
attribute version {
    xsd:decimal { minInclusive="1.1" fractionDigits="1" }
},
# A zone NS RRset MUST have at least two NS records
nservlet,
nservlet+,
# It MAY contain some DS records
ds*
}

# DNS-SEC records
ds = element ds {
    # rdata MUST contain
    # <Keytag> | <Algorithm> | <Digest type> | <Digest>
    # as per RFC 4034
    #
    element rdata { text }
}

# A single name server
nservlet = element nservlet {
    # An nservlet entry MUST contain a DNS FQDN
    # for a NS RR (RFC 1035)
    element fqdn { text }
}

Appendix B. Schema Definition for rDNS Queue Entries

The XML schema definition below, in RelaxNG Compact [RELAXNG] form is used for queue interaction operations.

default namespace = "http://download.research.icann.org/rq/1.0"

# A document MAY either be a single queue entry
# or a collection of queued entries
start = queue | queuelist

# A list of zone names for view only.
queuelist = element queuelist {
    attribute version {
        xsd:decimal { minInclusive="1.0" fractionDigits="0" }
    },
    queue*
}
# A single queued zone record
queue = element queue {
    # The zone record’s name, e.g., 10.in-addr.arpa
    attribute name { text },
    # The customer (optional); derived from known state.
    attribute cust { text }?,
    # The canonical URL for this queue record (optional)
    attribute href { xsd:anyURI }?,
    # The acknowledgement URL for this queue record (optional)
    attribute ack { xsd:anyURI }?,
    # The IP version of the address for the zone record (optional)
    attribute ipversion { "ipv4" | "ipv6" }?,
    # The state of the zone (optional); for a queue, it
    # SHOULD always be pending
    attribute state { "pending" }?,
    # The submitted timestamp (optional)
    attribute submitted { xsd:dateTime }?,
    # The HTTP method used to update
    attribute method { "PUT" | "DELETE" },
    # The schema version (1.0) (optional)
    attribute version {
        xsd:decimal { minInclusive="1.0" fractionDigits="1" }?
    }?,
    # A zone NS RRset must have at least two NS records
    nserver,
    nserver+,
    # It MAY contain some DS records
ds*
}

dns-sec records

ds = element ds {
    # rdata MUST contain Flags | Protocol | Algorithm | Public Key
    # as per RFC 4034
    element rdata { text }
}

# A single name server
nserver = element nserver {
    # An nserver entry MUST contain a DNS FQDN
    # for a NS RR (RFC 1035)
    element fqdn { text }
}
Acknowledgements

An XML schema was initially provided by APNIC; however, necessity required a branch, and as such a new namespace and schema have been created. Recognition goes to APNIC for prior efforts in this area.

The author acknowledges feedback from a collective made up of various RIR technical folk; however, heartfelt thanks goes to Anand Buddhdev of the RIPE-NCC and Robert Loomans of APNIC for being both alpha and beta testers and providing valuable feedback.

Author’s Address

Terry Manderson
ICANN

Email: terry.manderson@icann.org