DNSOP
Internet-Draft
Updates: 1034, 1035 (if approved)
Intended status: Standards Track
Expires: 17 September 2026

O. Sur篁
C. Vidal
E. Hunt
Internet Systems Consortium

16 March 2026

Parent-Centric Delegation Handling in DNS Resolvers
draft-sury-dnsop-parent-centric-resolver-01

Abstract

This document specifies a parent-centric behavioral model for DNS recursive resolvers, in which delegation decisions are always based on the NS RRset (or DELEG RRset) received from the parent side of a zone cut and are never overwritten by child-side NS data.

The parent-centric model eliminates the "two sources of truth" problem inherent in the current DNS delegation design, closes the Ghost Domain and Phoenix Domain attack vectors, provides deterministic behavior in the presence of parent/child NS mismatches, and enables resolvers to safely accept sibling (out-of-bailiwick) glue by scoping delegation information to individual zone cuts. It also provides the behavioral foundation required for deployment of the DELEG extensible delegation mechanism.

This document updates RFC 1034 and RFC 1035.

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 17 September 2026.

Table of Contents

1. Introduction

The Domain Name System (DNS) [RFC1034] [RFC1035] uses NS (Name Server) records to delegate authority over portions of the namespace from a parent zone to a child zone. A fundamental consequence of the current delegation model is that NS records exist in two places: at the delegation point in the parent zone (the "parent-side" NS RRset) and at the apex of the child zone (the "child-side" or "apex" NS RRset).

Historically, most recursive resolver implementations have followed a "child-centric" approach: when a referral is received from a parent zone, the resolver caches the parent-side NS RRset, but subsequently replaces it with the child-side apex NS RRset if one is observed in a response from the delegated zone. This replacement is performed on the assumption that the child zone's apex NS RRset is "more authoritative" than the parent's delegation point NS RRset.

This document specifies a "parent-centric" behavioral model for recursive resolvers, in which delegation decisions are always based on the NS RRset (or DELEG RRset) received from the parent side of a zone cut, and are never overwritten by child-side NS data. The parent-centric model eliminates an entire class of inconsistency and security problems inherent in the child-centric approach. It also provides the behavioral foundation needed for the deployment of extensible delegation mechanisms such as the DELEG record [I-D.ietf-deleg].

This document does not prescribe any particular cache architecture, data-structure layout, or lookup ordering for implementations. Conforming implementations are free to use any internal organization -- including a single unified cache -- that achieves the required behavior.

1.1. Terminology

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.

Parent-side NS RRset: The set of NS records present at a delegation point in the parent zone and optional glue records (A + AAAA) [RFC9471] required for the iterative clients to find the addresses of name servers that are contained within a delegated zone. These records are non-authoritative in the parent zone and are included in referral responses.

Child-side NS RRset: The set of NS records present at the apex of the delegated (child) zone. These records are authoritative within the child zone.

Delegation information: The data a resolver uses to determine which nameservers are responsible for a zone cut: nameserver names, associated addresses from the glue records, and (when DELEG [I-D.ietf-deleg] is supported) DELEG and DELEGPARAM parameters. This information originates from referral responses received from a parent zone’s nameservers.

Child-centric resolver: A resolver that replaces parent-side delegation information with child-side apex NS data when available.

Parent-centric resolver: A resolver that retains parent-side delegation information as the authoritative source for delegation decisions and does not overwrite it with child-side NS data.

Answer data: Resource records cached by the resolver for the purpose of answering client queries. In the context of this document, the child-side NS RRset is answer data: it is the authoritative answer to an NS query at a zone apex.

Delegation data: Information used internally by the resolver to determine which nameservers to contact when resolving queries. In the context of this document, the parent-side NS RRset (and associated glue) is delegation data. Delegation data is never returned to clients as an answer.

2. Problem Statement

2.1. Two Sources of Truth

[RFC1034] Section 4.2.1 specifies that NS records appear both at the delegation point in the parent zone and at the apex of the child zone. Ideally these two NS RRsets are identical, but in practice they frequently diverge. When a zone administrator updates the nameserver set, it is common for either the parent or the child NS RRset to be updated first, creating a window (sometimes indefinitely long) during which the two disagree.

A child-centric resolver that replaces parent-side NS data with child-side NS data effectively trusts the child zone to define its own delegation. This creates several problems:

* A compromised or misconfigured child zone can redirect resolution by publishing NS records pointing to servers not sanctioned by the parent.

* NS RRsets learned from the child side may include nameservers that are not present at the delegation point, creating inconsistencies in the resolver's view of the namespace. Or NS RRsets learned from the child side may not include nameserver that are present at the delegation point, again creating inconsistencies in the resolver's view of the namespace. It is up to the implementation whether the two distinct sets of NS RRsets get merged or whether the child-side NS RRset replace the parent-side delegation information.

* The child-side NS RRset may have a different (often longer) TTL than the parent-side delegation, causing the resolver to use stale delegation information. Alternative parent-side NS RRsets may have a different TTL than the child-side NS RRset. It is again up to the implementation whether the parent-side NS RRset TTL is honored, the child-side NS RRset TTL is honored or the smallest value of the two is chosen.

2.2. Ghost Domain Attacks

The Ghost Domain attack [GHOST-DOMAIN] demonstrated that a child-centric resolver's willingness to replace cached delegation data with fresher child-side NS records allows an attacker to keep a revoked domain name resolvable indefinitely. The attack works by exploiting the cache update logic: even after the parent zone has removed the delegation, the attacker can trigger queries that cause the resolver to contact the (still-running) rogue nameserver, which returns a child-side NS RRset with a fresh TTL, overwriting the stale (or

absent) parent-side delegation data in the cache. Experiments showed that more than 70% of tested open resolvers were still resolving a revoked domain a week after removal from the parent zone at the time the research was published.

The follow-up work on Phoenix Domain [PHOENIX-DOMAIN] showed that the original Ghost Domain mitigations (TTL capping on NS replacement) were insufficient. The Phoenix Domain attack uncovered additional vulnerable cache operations in the delegation process that allowed revoked domains to remain resolvable for over a month on more than 25% of tested resolvers at the time this research was published. Critically, the Phoenix Domain paper identifies the child-centric vs. parent-centric distinction as a key architectural factor: parent-centric resolvers that do not overwrite delegation data with child-side NS records are immune to the T1 class of Phoenix Domain attacks, because the attacker cannot inject fresh delegation data through child-zone responses.

A parent-centric resolver as specified in this document eliminates the Ghost Domain and Phoenix Domain T1 attack vectors entirely, because delegation decisions are never influenced by child-side NS data.

2.3. Inconsistent RPZ Behavior

Response Policy Zones (RPZ) [RFC7719] include NSDNAME and NSIP trigger types that match against the nameservers of the zone containing the answer. When a child-centric resolver replaces the parent-side NS RRset with the child-side NS RRset, the set of nameserver names and addresses visible to RPZ policy evaluation changes unpredictably depending on whether the child-side data has been cached yet. This leads to inconsistent policy enforcement: the same query may or may not trigger an RPZ rule depending on the cache state.

A parent-centric resolver provides a stable set of delegation information for RPZ evaluation, because the delegation data used for resolution is always the parent-side data received in referrals, regardless of what child-side NS data may exist in the cache.

2.4. Impediment to DELEG Deployment

The DELEG record [I-D.ietf-deleg] is an extensible delegation mechanism that is authoritative on the parent side of a zone cut. DELEG records can carry server addresses, transport parameters, and other extensible metadata directly in the delegation, eliminating the need for additional queries to resolve nameserver addresses. The companion DELEGPARAM record provides an indirection mechanism for

sharing delegation parameters across multiple zones.

A child-centric resolver cannot cleanly support DELEG, because:

* DELEG data is exclusively parent-side; there is no child-side equivalent. A child-centric resolver that overwrites parent-side data with child-side NS data would lose the DELEG information.

* [I-D.ietf-deleg] Section 5.1.1 requires that when DELEG records exist at a delegation point, a DELEG-aware resolver MUST use the name servers from those DELEG records and MUST NOT use NS records for the zone, even if resolution using DELEG records has failed. A parent-centric resolver naturally enforces this rule because it never overwrites parent-side delegation data with child-side NS data.

2.5. Interaction with Strict Glue Checking

Recent resolver releases have adopted strict (hardened) glue checking [STRICT-GLUE]: glue records in referral responses are only accepted when the NS target name is at or below the delegation point (in-domain glue). Glue for out-of-domain, but in-the-bailiwick nameserver names -- "sibling glue" -- is discarded, and those names must be resolved iteratively.

For example, given the delegation from .org:

```
example.org.          NS  ns1.example.org.
example.org.          NS  ns2.otherdomain.org.
ns1.example.org.      A   198.51.100.42
ns2.otherdomain.org.  A   203.0.113.53
```

strict glue checking accepts the A record for ns1.example.org (which is below example.org) but discards the A record for ns2.otherdomain.org (which is not). The resolver must then iteratively resolve ns2.otherdomain.org, following a separate delegation chain through the otherdomain.org zone.

This strict checking revealed a misconfiguration where a set of domains used cyclic in-the-bailwick glue and relied on the relaxed acceptance of the glue records for resolution.

For example, give the delegation from .org:

```
example.org.          NS  ns1.otherdomain.org.
example.org.          NS  ns2.otherdomain.org.
ns1.otherdomain.org.  A   198.51.100.42
ns2.otherdomain.org.  A   203.0.113.53

otherdomain.org.      NS  ns1.example.org.
otherdomain.org.      NS  ns2.example.org.
ns1.example.org.      A   198.51.100.42
ns2.example.org.      A   203.0.113.53
```

strict glue validation would throw away both set of glue records and make both example.org and otherdomain.org would fail to resolve.

2.5.1. Why Strict Glue Was Necessary

In a child-centric resolver that stores all cached data in a shared namespace, sibling glue is dangerous because it can contaminate unrelated delegations. If a referral for example.org includes a glue A record for ns1.example.net, that record enters the shared cache and may be used when resolving names under example.net -- even though the glue was provided by the .org zone, which has no authority over example.net addresses. An attacker who controls a delegation in .org can exploit this to inject forged address records for nameservers in .net, effectively poisoning the resolution of unrelated zones.

Strict glue checking prevents this cross-delegation contamination by rejecting all sibling glue. However, this strict policy causes significant operational problems:

* When sibling glue is rejected, the resolver performs more iterative lookups, each of which traverses additional delegation chains. Any parent/child NS mismatch along those chains can cause resolution failures or inconsistent behavior.

* In pathological cases, the combination of strict glue rejection and nested sibling delegations can produce resolution failures that succeed on retry (after intermediate records have been cached), making the problem intermittent and difficult to diagnose.

* The increased query volume puts additional load on authoritative servers and increases resolution latency for end users.

2.5.2. How Parent-Centric Behavior Resolves This

The parent-centric model eliminates the underlying security problem that strict glue checking was designed to address, because delegation information is self-contained per zone cut and is not shared across delegations.

When a parent-centric resolver accepts sibling glue for ns1.example.net in a referral for example.org, that address is recorded as part of the example.org delegation information. It is used only when contacting servers for example.org and is never used when resolving names under example.net or any other zone. The delegation information for example.net is a separate entry, populated from referrals received from the .net zone's parent, and is completely unaffected by glue received in referrals for example.org.

This scoping means that a parent-centric resolver can safely accept sibling glue again without risk of cross-delegation contamination. The operational problems caused by strict glue rejection -- increased query volume, resolution failures due to nested out-of-bailiwick delegations, and sensitivity to parent/child NS mismatches -- are avoided, because the sibling glue is used directly from the delegation information without requiring additional iterative resolution.

Resolver implementations adopting the parent-centric model MAY therefore accept sibling glue from referral responses, provided that the glue is stored as part of the delegation information for the specific zone cut and is not used for any other purpose.

3. Specification

3.1. Core Behavioral Requirement

A conforming resolver MUST use delegation information received from parent zones (via referral responses) as the basis for all delegation decisions. Delegation information MUST NOT be overwritten, replaced, or supplemented by child-side NS RRsets observed in authoritative responses from the delegated zone.

This requirement applies regardless of how the resolver organizes its internal data structures. Any particular cache architecture, data-structure choice, or lookup strategy is an implementation detail and out-of-the-scope for this document.

3.2. Processing Referrals

When a resolver receives a referral (a response with no answer, containing NS records in the authority section and optional glue in the additional section), it MUST process the delegation as follows:

1. Extract the zone cut name from the owner name of the NS RRset in the authority section.

2. Record the delegation information: the nameserver names from the NS RRset, and any A/AAAA glue records present in the additional section for those nameserver names. Because delegation information in the parent-centric model is scoped to the individual zone cut and is not shared across delegations, both in-domain glue and sibling glue can be safely accepted; see Section 2.5 for the security analysis. The effective TTL of the delegation information SHOULD be the minimum of the NS RRset TTL and the accepted glue record TTLs.

3. Use this delegation information for subsequent resolution of names under the zone cut, until it expires or is replaced by a newer referral for the same zone cut.

The resolver MUST NOT use the referral NS RRset to respond to the ordinary DNS clients. The referral NS RRset is parent-side data and MUST be treated as delegation information only.

3.3. Processing Authoritative Responses

When a resolver receives an authoritative response (AA bit set) that includes a child-side NS RRset (either in the answer section for an NS-type query, or in the authority section as part of the response), the resolver MUST cache the child-side NS RRset as answer data, subject to normal cache admission policies. This cached child-side NS RRset is used to answer explicit NS queries from clients (Section 3.4) but MUST NOT be used to update, replace, or supplement the delegation information used for delegation decisions.

The child-side NS RRset is authoritative data from the child zone and is the correct answer to an NS query. The parent-side NS RRset received in referrals is non-authoritative delegation information and MUST NOT be returned as the answer to an NS query.

3.4. Answering Explicit NS Queries

When a recursive resolver receives a query for the NS RRset at a zone name (QTYPE=NS, QNAME=<zone apex>), it MUST resolve and return the child-side (authoritative) NS RRset, not the parent-side delegation data. This is a fundamental distinction of the parent-centric model: parent-side NS data drives delegation decisions internally, but the child-side NS data is the authoritative answer to NS queries.

The resolver processes this query through normal recursive resolution:

1. Using its delegation information, the resolver contacts an authoritative server for the queried zone.

2. The authoritative server returns the apex NS RRset with the AA (Authoritative Answer) flag set.

3. The resolver caches this child-side NS RRset as answer data and returns it to the client.

The response to the client MUST accurately reflect the properties of the child-side NS data:

AA flag: The child-side NS RRset is authoritative data from the child zone. When a resolver that is also configured as an authoritative server for the queried zone returns this data, it MUST set the AA flag. When a purely recursive resolver returns a cached child-side NS RRset, it SHOULD set the AA flag to indicate that the data originates from the authoritative child zone, distinguishing the response from a referral (which would contain parent-side NS data without the AA flag). This is consistent with the principle that the answer to an NS query at a zone apex is authoritative data, regardless of how the resolver obtained it.

AD flag: If the child-side NS RRset has been DNSSEC-validated, the resolver SHOULD set the AD (Authenticated Data) flag in the response, subject to the normal AD flag rules ([RFC4035] Section 3.2.3 and [RFC6840] Section 5.8). DNSSEC validation of the child-side NS RRset provides the client with cryptographic assurance that the nameserver set is authentic, a property that parent-side delegation NS data (which is unsigned glue) can never provide.

The resolver MUST NOT return the parent-side delegation NS data as the answer to an NS query under any circumstances. The parent-side data is non-authoritative and would provide the client with incorrect information about the zone’s published nameserver set.

Note that the child-side NS RRset returned to the client may differ from the parent-side NS RRset used for delegation. This is expected and correct: the two serve different purposes. The child-side NS is the zone's own statement of its nameservers (answer data), while the parent-side NS is the parent's statement of which servers are delegated (delegation data).

3.4.1. Interaction with Minimal Responses

When QNAME minimization [RFC9156] is enabled (as RECOMMENDED for cached responses in Section 5.2), the resolver omits the NS RRset from the authority section of non-NS-type responses. This has no effect on explicit NS queries: when the client queries for NS records specifically, the full child-side NS RRset is returned in the answer section regardless of the minimal responses setting.

3.4.2. NS Queries When Child-Side NS Is Not Cached

If the resolver receives an NS query for a zone name and does not have the child-side NS RRset cached, it MUST resolve the query through normal recursion (contacting the zone's authoritative servers) rather than returning the parent-side delegation data from its delegation information. The parent-side data is not a valid answer to an NS query.

3.5. Best Zone Cut Selection

When a resolver needs to determine the closest enclosing zone cut for a query name (the "best zone cut"), it MUST select among the following sources in this order of preference:

1. Locally configured zones: forward zones, stub zones, static-stub zones, and other policy-configured delegations.

2. Delegation information learned from referrals, as described in Section 3.2.

3. Root hints, if no other delegation information is available.

When both a locally configured zone and referral-learned delegation information exist for overlapping names, the more specific (closer to the query name) delegation is preferred, with the exception that static-stub zones always take precedence over referral-learned delegations at the same name.

Child-side NS RRsets cached from authoritative responses (Section 3.3) MUST NOT be considered when selecting the best zone cut.

3.6. TTL and Expiration

Delegation information learned from referrals is subject to TTL-based expiration. The TTL is computed as described in Section 3.2. Expired delegation information MUST NOT be used for delegation decisions; the resolver MUST fall back to the next available delegation source (a less specific referral-learned delegation, a locally configured zone, or root hints).

Implementations SHOULD NOT apply neither a minimum TTL floor or a maximum TTL ceiling to delegation information that differs from the floor applied to cached answer data, as the two serve different purposes and have different operational characteristics. The implementations MAY have a seperate configuration options to apply the limits to the TTL received for the delegation information.

3.7. Delegation Information Replacement

When a resolver receives a new referral for a zone cut for which it already holds unexpired delegation information, the new referral replaces the existing delegation information. This ensures that the resolver tracks changes in the parent zone’s delegation.

A resolver MUST NOT merge delegation information from multiple referrals for the same zone cut. Each referral provides a complete delegation, and the most recently received delegation supersedes any previous one.

3.8. DELEG Integration

When a resolver supports the DELEG record [I-D.ietf-deleg], delegation information for a zone cut may include DELEG parameters (server addresses, server names, DELEGPARAM references, transport parameters) in addition to or instead of NS-based delegation data.

A conforming resolver that receives a referral containing DELEG records at a zone cut MUST use the DELEG-based delegation information to populate its delegation information (per [I-D.ietf-deleg] Section 5.1.3 and Section 5.1.4) and MUST NOT use NS records for that zone, even if DELEG-based resolution fails (per [I-D.ietf-deleg] Section 5.1.1).

The parent-centric model naturally accommodates DELEG because:

* DELEG data is exclusively parent-side and arrives in referrals, which is exactly the data the parent-centric resolver trusts.

* DELEG records can carry server addresses directly (via server-ipv4 and server-ipv6 parameters), eliminating the need for glue records and the associated trust questions. DELEGPARAM records can provide an additional layer of indirection for shared delegation configurations.

* DELEG records are signed on the parent side, enabling DNSSEC validation of delegation parameters -- a capability fundamentally impossible with unsigned parent-side NS records.

4. Interaction with Other Mechanisms

4.1. QNAME Minimization

QNAME minimization [RFC9156] relies on knowing the current best zone cut in order to construct minimized queries. A parent-centric resolver provides more reliable zone cut information for QNAME minimization because the delegation data used for zone cut determination is never polluted by child-side NS data that might indicate a different (and potentially incorrect) zone boundary.

When resuming QNAME minimization after receiving a referral, the resolver uses the updated delegation information as the basis for subsequent minimized queries.

4.2. DNSSEC Validation

Parent-side NS records are not signed (they are glue), so delegation information learned from NS-based referrals is not directly DNSSEC-validated. This is consistent with current practice: even child-centric resolvers use unsigned referral NS data to locate authoritative servers.

The DNSSEC chain of trust for a delegated zone is established through DS records, which are authoritative on the parent side and are validated normally. The parent-centric model does not change DS record processing.

When DELEG records are deployed (Section 3.8), the delegation parameters are signed on the parent side and can be validated as part of the delegation. This provides a cryptographic binding between the delegation parameters and the parent zone’s DNSSEC chain -- a significant security improvement over the current unsigned NS-based delegation model.

4.3. Delegation Limits

Resolvers typically impose a limit on the number of nameservers or addresses they will process from a single delegation, to prevent excessive resource consumption from very large delegations. The parent-centric model preserves this limit. Implementations SHOULD make this limit configurable with a reasonable default (e.g., 26). Q?: is 2 * 13 good number?

5. Operational Considerations

5.1. Compatibility with Existing Infrastructure

The parent-centric model is fully compatible with the existing DNS infrastructure. No changes are required to authoritative servers, registries, registrars, or stub resolvers. The change is entirely within the recursive resolver’s internal processing.

This approach has been deployed in production by multiple resolver implementations, including Nominum Vantio and Google Public DNS, without causing widespread operational issues.

More implementations here???

5.2. Impact on Responses to Recursive Clients

A child-centric resolver that has both parent-side and child-side NS data may include the child-side NS in the authority section of responses to recursive clients (when minimal responses option is disabled). A parent-centric resolver that does not use child-side NS data for delegation has no reason to look up the child-side NS RRset during resolution; therefore, the child-side NS may not be readily available for inclusion in the authority section.

Implementations adopting the parent-centric model SHOULD treat recursive responses served from the cache as if minimal response option is enabled, omitting the NS RRset from the authority section. This is consistent with modern best practice, reduces response size, and avoids the need to perform additional lookups solely for populating the authority section.

This only affects responses served from the cache by a recursive resolver. Authoritative responses (where the server is authoritative for the queried zone) are not affected by this change.

TODO: I am actually not sure if this is a correct mechanism or not :shrug:. Alternative proposal below, but since this only impact the partial forwarders, I would say - it is a reasonable compromise to make...

When a recursiver resolver receives a query with RD flag not set (RD=0) for the NS RRset at a zone name (QTYPE=NS, QNAME=<delegation point>), it MUST resolve and return the parent-side (delegation) NS RRset, not the child-side (authoritative) NS RRset.

There might be downstream forwarding resolvers (e.g. DNS resolver using this resolver for DNS resolution). Returning child-side NS RRset is generally not a problem when the downstream DNS resolver is configured to do the forwarding for the whole DNS tree, but there might be configurations where the forwarding is configured only for certain parts of the DNS tree and the downstream DNS forwarder need to receive the parent-side delegation information.

5.3. Zones with Differing Parent and Child NS Sets

Some zones intentionally maintain different NS RRsets at the parent and child, for example to include "stealth" nameservers in the child-side NS RRset that are not registered at the parent. Under the parent-centric model, these stealth nameservers will not be used for delegation decisions. This is the intended and correct behavior: the parent zone is the authority for delegation, and nameservers not listed in the parent’s delegation are not part of the delegation. Also this behavior was already breaking the existing DNS standards, the parent-side and child-side NS RRset is supposed to be in sync.

The child-side NS RRset remains available for answering explicit NS queries (Section 3.4) and for any purpose other than delegation decisions. Clients that query for the NS RRset at a zone apex will receive the child-side NS data (with the AA flag set), which may include nameservers not present in the parent-side delegation. This makes the difference between the two NS sets visible and auditable by zone operators.

5.4. Transition Considerations

Resolver operators transitioning from a child-centric to a parent-centric implementation should be aware of the following:

* After the transition, responses to queries with RD=1 may no longer include NS records in the authority section for cached answers. Monitoring systems that check for the presence of authority-section NS records in recursive responses should be updated.

* Zones that rely on child-side NS records to "override" the parent's delegation (e.g., to work around stale delegation data at a registry) will no longer benefit from this override. Such zones should instead update their parent-side delegation.

* RPZ rules that were tuned based on child-side NS data may need to be reviewed, as the nameserver set visible to RPZ evaluation will be the (more stable) parent-side set.

* Resolver implementations that have previously deployed strict glue checking (rejecting sibling glue) may relax this restriction after adopting the parent-centric model, because delegation information is scoped per zone cut and sibling glue can no longer contaminate unrelated delegations (see Section 2.5). Accepting sibling glue again reduces the number of iterative queries needed to resolve out-of-bailiwick delegations and avoids the resolution failures associated with strict glue rejection.

6. Security Considerations

6.1. Elimination of Child-Side NS Overwrite Attacks

The most significant security benefit of the parent-centric model is that a compromised or malicious child zone cannot redirect resolution by publishing rogue NS records at its apex. Since the resolver never uses child-side NS data for delegation decisions, such attacks are ineffective.

This closes an attack vector that has been present in child-centric resolvers since the inception of the DNS: an attacker who compromises only the child zone (but not the parent) can add NS records pointing to attacker-controlled servers, and a child-centric resolver will follow those NS records for subsequent queries.

6.2. Mitigation of Ghost Domain and Phoenix Domain Attacks

As discussed in Section 2.2, the Ghost Domain [GHOST-DOMAIN] and Phoenix Domain [PHOENIX-DOMAIN] attacks exploit the child-centric cache update logic to keep revoked domain names resolvable. A parent-centric resolver is immune to the T1 class of these attacks because:

* It does not accept child-side NS RRsets as delegation data, so an attacker cannot refresh a stale or removed delegation by responding with a child-side NS RRset carrying a new TTL.

* Once the parent-side delegation information expires (or is never received because the parent has removed the delegation), the resolver falls back to the next enclosing zone cut or root hints, which will not lead to the revoked zone.

The T2 class of Phoenix Domain attacks (which exploits protocol-level TTL semantics in referral responses from still-delegated parent zones) is not fully addressed by the parent-centric model alone, but the attack surface is significantly reduced because the resolver maintains a clear separation between delegation data and answer data.

6.2.1. Elimination of Cross-Delegation Glue Contamination

In a child-centric resolver with a shared cache, the primary risk of sibling glue is cross-delegation contamination: an A/AAAA record for ns1.example.net received in a referral for example.org enters the shared cache and can be used when resolving example.net, even though the glue was provided by the .org zone which has no authority over example.net addresses. This is the attack vector that strict glue checking [STRICT-GLUE] was designed to prevent.

In the parent-centric model, delegation information is self-contained per zone cut. Glue received in a referral for example.org is part of the example.org delegation and is used only for contacting servers for that zone. It has no effect on the delegation information for example.net or any other zone. This scoping eliminates the cross-delegation contamination vector entirely, without the operational cost of discarding sibling glue.

A parent-centric resolver can therefore safely accept sibling glue from referral responses (see Section 2.5), recovering the operational benefits of sibling glue (fewer iterative queries, faster resolution, tolerance of out-of-bailiwick delegation configurations) while maintaining security against cross-delegation poisoning.

6.2.2. Remaining Glue Trust Considerations

Even with per-delegation scoping, glue records are still non-authoritative data that can be forged by an on-path attacker between the parent zone’s authoritative server and the resolver. A forged glue record would cause the resolver to contact an attacker-controlled server for the specific delegation that received the forged glue. This risk is identical to the existing risk of forged referral responses and is mitigated by the same mechanisms (source port randomization, DNS cookies, DNSSEC validation of the subsequent responses from the delegation).

When DELEG records are used, server addresses included directly in the DELEG RDATA are signed on the parent side, providing DNSSEC-validated address information. This eliminates the glue trust problem entirely for DELEG-aware delegations, as the addresses are authenticated data rather than unsigned glue, and out-of-bailiwick resolution of nameserver names is no longer necessary.

6.3. Stable Delegation View

By ensuring that delegation decisions are always based on parent-side data, the parent-centric model provides a more stable and predictable view of the delegation hierarchy. This stability benefits security mechanisms that depend on knowing the authoritative server set, including RPZ, DNS firewall policies, and logging/auditing systems.

7. IANA Considerations

This document has no IANA actions.

8. Implementation Status

> *Note to the RFC Editor:* Please remove this section before publication.

The parent-centric resolver behavior described in this document has been implemented in BIND 9 as part of a resolver refactoring. The implementation ensures that delegation decisions are based on referral-learned data rather than child-side NS RRsets from the cache. The fetch context uses a delegation set abstraction (containing nameserver names, IPv4/IPv6 addresses, and DELEG/DELEGPARAM parameters) instead of raw NS rdatasets throughout the resolution process.

The implementation passes the full BIND 9 system test suite with adjustments to tests that previously depended on child-side NS records appearing in the authority section of recursive responses.

The parent-centric approach has also been deployed in production by Google Public DNS and was previously used by Nominum Vantio.

More implementations?

9. References

9.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/rfc/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/rfc/rfc1035>.

[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/rfc/rfc2119>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/rfc/rfc4035>.

[RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, DOI 10.17487/RFC6840, February 2013, <https://www.rfc-editor.org/rfc/rfc6840>.

[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/rfc/rfc8174>.

[RFC9156] Bortzmeyer, S., Dolmans, R., and P. Hoffman, "DNS Query Name Minimisation to Improve Privacy", RFC 9156, DOI 10.17487/RFC9156, November 2021, <https://www.rfc-editor.org/rfc/rfc9156>.

[RFC9471] Andrews, M., Huque, S., Wouters, P., and D. Wessels, "DNS Glue Requirements in Referral Responses", RFC 9471, DOI 10.17487/RFC9471, September 2023, <https://www.rfc-editor.org/rfc/rfc9471>.

9.2. Informative References

[GHOST-DOMAIN] Jiang, J., Liang, J., Li, K., Li, J., Duan, H., and J. Wu, "Ghost Domain Names: Revoked Yet Still Resolvable", In Proceedings of the Network and Distributed System Security Symposium (NDSS 2012), February 2012, <https://www.ndss-symposium.org/ndss2012/>.

[I-D.ietf-deleg]
paek, P., Weber, R., and Lawrence, "Extensible Delegation for DNS", Work in Progress, Internet-Draft, draft-ietf-deleg-07, 6 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-deleg-07>.

[PHOENIX-DOMAIN]
Li, X., Liu, B., Bai, X., Duan, H., Li, Q., and Q. Pan, "Ghost Domain Reloaded: Vulnerable Links in Domain Name Delegation and Revocation", In Proceedings of the Network and Distributed System Security Symposium (NDSS 2023), February 2023, <https://www.ndss-symposium.org/ndss-paper/ghost-domain-reloaded-vulnerable-links-in-domain-name-delegation-and-revocation/>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <https://www.rfc-editor.org/rfc/rfc7719>.

[STRICT-GLUE]
Internet Systems Consortium, "Operational Notification: Impact of Stricter Glue Checking", December 2025, <https://kb.isc.org/docs/strict-glue>.

Appendix A. Rationale for Delegations in Positive Responses

During implementation, the question arose of whether delegation information should also be recorded when it appears in positive authoritative responses (AA set) rather than only in referrals. Consider the following zone structure:

zone	nameserver
example	ns1
foo.example	ns2
bar.foo.example	ns2

To resolve a.bar.foo.example, the resolver:

1. Asks a root server for example, receives a referral to ns1.

2. Asks ns1 for foo.example, receives a referral to ns2.

3. Asks ns2 for bar.foo.example/NS, receives a positive authoritative answer (AA set) with the NS in the answer section.

If the delegation from step 3 is not recorded, the closest zone cut known to the resolver is foo.example, which still works because ns2 is authoritative for both foo.example and bar.foo.example.

However, not recording this delegation means the resolver will not have address information for bar.foo.example nameservers available, so additional section data in subsequent responses may be incomplete. Whether this trade-off is acceptable depends on the deployment context and whether minimal responses options is in use.

Implementations MAY choose to record delegation information learned from positive authoritative responses, but SHOULD carefully evaluate the impact on query volume and consistency.

Appendix B. Relationship to DELEG

The DELEG record [I-D.ietf-deleg] defines an extensible delegation mechanism that is exclusively parent-side. The parent-centric resolver behavior described in this document provides the necessary behavioral foundation for DELEG support:

* Delegation information as defined in this document naturally accommodates DELEG data alongside or instead of NS-based delegations.

* Each delegation can contain DELEG entries with embedded server addresses (server-ipv4, server-ipv6), server names (server-name), and DELEGPARAM references (include-delegparam), in addition to traditional NS names and glue.

* When DELEG records are present at a delegation point, the parent-centric model provides a natural enforcement point for the rule from [I-D.ietf-deleg] Section 5.1.1 that a DELEG-aware resolver MUST use name servers from DELEG records and MUST NOT fall back to NS records.

* Because DELEG is signed on the parent side, a parent-centric resolver can validate delegation parameters through the normal DNSSEC chain of trust -- a capability that is fundamentally impossible with unsigned parent-side NS records.

The parent-centric behavioral model is a prerequisite for correct DELEG implementation. A child-centric resolver that overwrites parent-side delegation data with child-side NS data would lose the DELEG information and could not maintain the invariant that DELEG takes precedence over NS.

Appendix C. Intellectual Property Note

Note to the RFC Editor: Please remove this appendix before publication.

The authors are aware of U.S. Patent 7,769,826 ("Systems and methods of providing DNS services using separate answer and referral caches", filed June 26, 2003, assigned to Nominum, Inc., now held by Akamai Technologies). That patent claims a specific cache architecture in which answer information and referral information are stored in separate data structures (a flat/hash table for answers and a hierarchical/tree structure for referrals) with a particular lookup order and classification-based routing of responses to the appropriate store.

This document intentionally does not specify, recommend, or require any particular cache architecture, data-structure choice, lookup ordering, or response classification scheme. It specifies only the behavioral requirement that delegation decisions be based on parent-side NS (or DELEG) data from referrals and not be overwritten by child-side NS data. Conforming implementations are free to use any internal organization -- including a single unified cache -- that achieves the required behavior.

The authors believe that this document is not Covered by the above mentioned patent.

Acknowledgements

The concept of parent-centric delegation handling has been explored by multiple DNS implementations over the years. The authors would like to acknowledge the prior work by Nominum (Vantio) and Google (Public DNS) in deploying parent-centric resolvers at scale, which demonstrated the viability of this approach.

The BIND 9 implementation was developed by Colin Vidal with contributions from Evan Hunt and Ondej Sur, building on the issue analysis by Petr paek.

Authors' Addresses

Ondej Sur
Internet Systems Consortium
Czechia
Email: ondrej@isc.org

Colin Vidal
Internet Systems Consortium
France
Email: colin@isc.org

Evan Hunt
Internet Systems Consortium
United States of America
Email: each@isc.org