The Domain Name System standard mechanisms for maintaining coherent servers for a zone consist of three elements. One mechanism is the Authoritative Transfer (AXFR) defined in "Domain Names - Concepts and Facilities" [RFC1034] (referred to in this document as RFC 1034) and "Domain Names - Implementation and Specification" [RFC1035] (aka RFC 1035). Incremental Zone Transfer (IXFR) is defined in "Incremental Zone Transfer in DNS" [RFC1995]. A mechanism for prompt notification of zone changes (NOTIFY) is defined in "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)" [RFC1996]. The goal of these mechanisms is to enable a set of DNS name servers to remain coherently authoritative for a given zone.

Comments on this draft ought to be addressed to the editor or to namedroppers@ops.ietf.org.

1.1. Definition of Terms

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 1484. "Newer"/"New" DNS and "older"/"old" DNS refers to implementations written after and prior to the publication of this document.

1.2. Scope

In the greater context there are many ways to achieve coherency among a set of name servers. The AXFR, IXFR and NOTIFY mechanisms form just one, the one defined in the RFCs cited. For example, there are DNS implementations that assemble answers from data stored in relational databases (as opposed to master files) relying on the database's non-DNS means to synchronize the database instances. Some of these non-DNS solutions interoperate in some fashion. As far as it is known, AXFR, IXFR and NOTIFY are the only in-band mechanisms that provide an interoperable solution to the desire for coherency within the definition of DNS, they certainly are the only mechanisms documented by the IETF.

This document does not cover incoherent DNS situations. There are applications of the DNS in which servers for a zone are designed to be incoherent. For these configurations, a coherency mechanism as described here would be unsuitable.

"General purpose DNS implementation" refers to DNS software developed for wide-spread use. This includes resolvers and servers freely accessible as libraries and standalone processes. This also includes proprietary implementations used only in support of DNS service offerings.

"Turnkey DNS implementation" refers to custom made, single use implementations of DNS. Such implementations consist of software that employs the DNS protocol message format yet do not conform to the entire range of DNS functionality.

A DNS implementation is not required to support AXFR, IXFR and NOTIFY. A DNS implementation SHOULD have some means for maintaining name server
coherency. A general purpose DNS implementation SHOULD include AXFR (and in the same vein IXFR and NOTIFY), but turnkey DNS implementations MAY exist without AXFR. (An editorial note to readers of this section. The mention of IXFR and NOTIFY is for context and illustration, this document does not make any normative comments on those mechanisms.)

### 1.3 Context

Besides describing the mechanisms themselves, there is the context in which they operate to consider. When AXFR, IXFR and NOTIFY were defined, there was little consideration given to security and privacy issues. Since the original definition of AXFR, new opinions have appeared on the access to an entire zone’s contents. In this document, the basic mechanisms will be discussed separately from the permission to use these mechanisms.

### 1.4 Coverage

This document concentrates on just the definition of AXFR. Any effort to update the IXFR or NOTIFY mechanisms would be done in different documents. This is not strictly a clarification of the definition in RFC 1034 and RFC 1035. This document will update those sections, and invalidate at least one part of that definition. The goal of this document is to define AXFR as it exists, or is supposed to exist, currently.

### 1.4 Coverage and Relationship to Original AXFR Specification

This document concentrates on just the definition of AXFR. Any effort to update the IXFR or NOTIFY mechanisms would be done in different documents.

The original "specification" of the AXFR sub-protocol is scattered through RFC 1034 and RFC 1035. Section 2.2 of RFC 1035 on page 5 depicts the scenario for which AXFR has been designed. Section 4.3.5 of RFC 1034 describes the zone synchronization strategies in general and rules for the invocation of a full zone transfer via AXFR; the fifth paragraph of that section contains a very short sketch of the AXFR protocol. Section 3.2.3 of RFC 1035 has assigned the code point for the AXFR QTYPE (see section 2.1.2 below for more details). Section 4.2 of RFC 1035 discusses the transport layer use of DNS and shortly explains why UDP transport is deemed inappropriate for AXFR; the last paragraph of Section 4.2.2 gives details for the TCP connection management with AXFR. Finally, the second paragraph of Section 6.3 in RFC 1035 mandates server behavior when zone data changes occur during an ongoing zone transfer using AXFR.

This document will update the specification of AXFR in fully specifying the record formats and processing rules for AXFR, largely expanding on paragraph 5 of Section 4.3.5 of RFC 1034, and detailing the transport considerations for AXFRs, thus amending Section 4.2.2 of RFC 1035. Furthermore, it discusses backward compatibility issues and provides policy/management considerations as well as specific Security Considerations for AXFR. The goal of this document is to define AXFR as it exists, or is supposed to exist, currently.

### 2 AXFR Messages

An AXFR session consists of an exchange of an AXFR query message and a set of AXFR response messages. In this document, the AXFR client is the sender of the AXFR query and the AXFR server is the responder. (Use of terms such as master, slave, primary, secondary are not important to defining AXFR.) The use of the word "session" without qualification refers to an AXFR session.

An important aspect to keep in mind is that the definition of AXFR is restricted to TCP [RFC0793]. The design of the AXFR process has certain inherent features that are not easily ported to UDP [RFC0768].

The basic format of an AXFR message is the DNS message as defined in RFC 1035, Section 4 ("MESSAGES") [RFC0235], updated by the following:

- "A Mechanism for Prompt Notification of Zone Changes (...)" [RFC1996]
- "Domain Name System (DNS) IANA Considerations" [RFC5095]
- "Dynamic Updates in the Domain Name System (DNS UPDATE)" [RFC2136]
- "Extension Mechanisms for DNS (EDNS)" [RFC2626]
- "Secret Key Transaction Authentication for DNS (TSIG)" [RFC2845]
- "Secret Key Establishment for DNS (TKEY RR)" [RFC2930]
- "Obsoleting IQQUIP" [RFC3420]
- "Handling of Unknown DNS Resource Record (RR) Types" [RFC3597]
- "Protocol Modifications for the DNS Security Extensions" [RFC4035]
- "GMAC SHA TIG Algorithm Identifiers" [RFC4035]

Field names used in this document will correspond to the names as they appear in the IANA registry for DNS Header Flags [DNSFLGS].

#### 2.1 AXFR query

An AXFR query is sent by a client whenever there is a reason to ask. This might be because of zone maintenance activities or as a result of a command line request, say for debugging.

##### 2.1.1 Header Values

These are the DNS message header values for an AXFR query.

- **ID**: See note 2.1.1.a
- **QR**: MUST be 0 (Query)
- **OPCODE**: MUST be 0 (Standard Query)
- **AA**: See note 2.1.1.b
- **RD**: See note 2.1.1.b
- **RA**: See note 2.1.1.b
- **Z**: See note 2.1.1.c
- **AD**: See note 2.1.1.b
- **CD**: See note 2.1.1.b
- **RCODE**: MUST be 0 (No error)
QDCOUNT MUST be 1
ANCOUNT MUST be 0
NSCOUNT MUST be 0
ARCOUNT See note 2.1.1.d

Note 2.1.1.a Set to any value that the client is not already using with the same server. There is no specific means for selecting the value in this field. (Recall that AXFR is done only via TCP connections.)

A server MUST reply using messages that use the same message ID to allow a client to meaningfully have multiple AXFR queries outstanding.

Note 2.1.1.b The value in this field has no meaning in the context of AXFR query messages. For the client, it is recommended that the value be zero. The server MUST ignore this value.

Note 2.1.1.c The client MUST set this bit to 0, the server MUST ignore it.

Note 2.1.1.d The client MUST set this field to be the number of resource records appearing in the additional section. See Section 2.1.5 "Additional Section" for details.

The value MAY be 0, 1 or 2. If it is 2, the additional section MUST contain both an EDNS0 [RFC2671] OPT resource record and a record carrying transaction integrity and authentication data, currently a choice of TSIG [RFC2845] and SIG(0) [RFC2931]. If the value is 1, then the additional section MUST contain either only an EDNS0 OPT resource record or a record carrying transaction integrity and authentication data. If the value is 0, the additional section MUST be empty.

2.1.2 Query Section

The Query section of the AXFR query MUST conform to section 4.1.2 of RFC 1035, and contain the following values:

QNAME the name of the zone requested
QTYPE AXFR (= 252), the pseudo-RR type for zone transfer [DNSVALS]
QCLASS the class of the zone requested

2.1.3 Answer Section

MUST be empty.

2.1.4 Authority Section

MUST be empty.

2.1.5 Additional Section

The client MAY include an EDNS0 OPT [RFC2671] resource record. If the server has indicated that it does not support EDNS0, the client MUST send this section without an EDNS0 OPT resource record if there is a retry. Indication that a server does not support EDNS0 is not an explicit element in the protocol, it is up to the client to interpret. Most likely, the server will return a FORMERR which might be related to the OPT resource record.

The client MAY include a transaction integrity and authentication resource record, currently a choice of TSIG [RFC2845] or SIG(0) [RFC2931]. If the server has indicated that it does not recognize the resource record, and that the error is indeed caused by the resource record, the client probably ought not try again. Removing the security data in the face of an obstacle ought to only be done with full awareness of the implication of doing so.

In general, if an AXFR client is aware that an AXFR server does not support a particular mechanism, the client SHOULD NOT attempt to engage the server using the mechanism (or at all). A client could become aware of a server's abilities via a configuration setting or via some other (as yet) undefined means.

The range of permissible resource records that MAY appear in the additional section might change over time. If either a change to an existing resource record (like the OPT RR for EDNS0) is made or a new additional section record is created, the new definitions ought to include a discussion on the impact upon AXFR. Although this is not predictable, future additional section residing records may have an effect that is orthogonal to AXFR, so can ride through the session as opaque data. In this case, a "wise" implementation ought to be able to pass these records through without disruption.

2.2 AXFR response

The AXFR response will consist of 0 or more messages. A "0 message" response is covered in section 2.2.1.

An AXFR response that is transferring the zone's contents will consist of a series (which could be a series of length 1) of DNS messages. In such a series, the first message MUST begin with the SOA resource record of the zone, the last message MUST conclude with the same SOA resource record. Intermediate messages MUST NOT contain the SOA resource record. The first message MUST copy the Query Section from the corresponding AXFR query message into its Query Section. The inclusion of the terminating SOA resource record is not necessary.

An AXFR response that is indicating an error MUST consist of a single DNS message with the return code set to the appropriate value for the condition encountered - once the error condition is detected. Such a message MUST copy the AXFR query Query Section into its Query Section. The inclusion of the terminating SOA resource record is not necessary.

An AXFR client might receive a number of AXFR response messages free of an error condition before the message indicating an error is received. But once an error is reported, the AXFR client can assume that the error reporting message is the last message sent by the AXFR server in the current AXFR session.

2.2.1 "0 Message" Response
A legitimate "0 message" response, i.e., the client sees no response whatever, is very exceptional and controversial. Unquestionably it is unhealthy for there to be 0 responses in a protocol that is designed around a query - response paradigm over an unreliable transport. The lack of a response could be a sign of underlying network problems and cause the protocol state machine to react accordingly. However, AXFR uses TCP and not UDP, eliminating undetectable network errors.

A "0 message response" is reserved for situations in which the server has a reason to suspect that the query is sent for the purpose of abuse. Due to the use of this being so controversial, a "0 message response" is not being defined as a legitimate part of the protocol but the use of it is being acknowledged as a warning to AXFR client implementations. Any earnest query has the expectation of some response but may not get one.

### 2.2.2 Header Values

| ID | See note 2.2.2.a |
| QR | MUST be 1 (Response) |
| Opcode | MUST be 0 (Standard Query) |
| AA | See note 2.2.2.b |
| TC | MUST be 0 (Not truncated) |
| RD | RECOMMENDED copy request's value, MAY be set to 0 |
| RA | See note 2.2.2.c |
| AD | See note 2.2.2.d |
| CD | See note 2.2.2.e |
| RD | See note 2.2.2.f |
| QDCOUNT | MUST be 1 in the first message; MUST be 0 or 1 in all following |
| ANCOUNT | MUST be 0 |
| ARCOUNT | See note 2.2.2.g |
| ANSWER | See note 2.2.2.h |

Note 2.2.2.a Because some old implementations behave differently than is now desired, the requirement on this field is stated in detail. New DNS servers MUST set this field to the value of the AXFR query ID in each AXFR response message for the session. AXFR clients MUST be able to manage sessions resulting from the issuance of multiple outstanding queries, whether AXFR queries or other DNS queries. A client SHOULD discard responses that do not correspond (via the message ID) to any outstanding queries.

Note 2.2.2.b If the RCODE is 0 (no error), then the AA bit MUST be 1. For any other value of RCODE, the AA bit MUST be set according to rules for that error code. If in doubt, it is RECOMMENDED that it be set to 1. It is RECOMMENDED that the value be ignored by the AXFR client.

Note 2.2.2.c It is RECOMMENDED that the server set the value to 0, the client MUST ignore this value.

Note 2.2.2.d The server MUST set this bit to 0, the client MUST ignore it.

Note 2.2.2.e If the implementation supports the DNS Security Extensions (see below) then this value MUST be set according to the rules in RFC 4035, section 3.1.6, "The AD and CD Bits in an Authoritative Response". If the implementation does not support the DNS Security Extensions, then this value MUST be set to 0 and MUST be ignored upon receipt.

The DNS Security Extensions (DNSSEC) is defined in these base documents:

- "DNS Security Introduction and Requirements" [RFC4033]
- "Resource Records for the DNS Security Extensions" [RFC4034]
- "Protocol Modifications for the DNS Security Extensions" [RFC4035]

Note 2.2.2.f In the absence of an error, the server MUST set the value of this field to NoError. If a server is not authoritative for the queried zone, the server SHOULD set the value to NotAuth. (Reminder, consult the appropriate IANA registry (DNSVALS:).) If a client receives any other value in response, it MUST act according to the error. For example, a malformed AXFR query or the presence of an EDNS0 OPT resource record sent to an old server will garner a FormErr value. This value is set as part of the AXFR response processing. The same is true for other error-indicating values.

Note 2.2.2.g The count of answer records MUST equal the number of resource records in the AXFR Answer Section. When a server is aware that a client will only accept one resource record per response message, then the value MUST be 1. A server MAY be made aware of a client's limitations via configuration data.

Note 2.2.2.h The client MUST set this field to the number of resource records appearing in the additional section. See Section 2.1.1 "Additional Section" for details.

### 2.2.3 Query Section

In the first response message, this section MUST be copied from the query. In subsequent messages, this section MAY be copied from the query if it MAY be empty. The content of this section MAY be used to determine the context of the message, that is, the name of the zone being transferred. If the server receives any other value in response, it MUST act according to the error.
encoding zone contents.

2.2.5 Authority Section

MUST be empty.

2.2.6 Additional Section

The contents of this section MUST follow the guidelines for EDNS0, TSIG, SIG(0), or what ever other future record is possible here. The contents of section 2.1.5 apply here as well.

Note that TSIG and SIG(0), if in use, will treat each individual AXFR response message within a session as a unit of data. That is, each message will have a TSIG or SIG(0) (if in use) and the cryptographic check will cover just that message. The same rule will apply to future alternatives and documents covering them ought to consider the impact on AXFR response messages.

2.3 TCP Connection Aborts

If an AXFR client sends a query on a TCP connection and the connection is closed at any point, the AXFR client MUST consider the AXFR session terminated. The message ID MAY be used again on a new connection, even if the question and AXFR server are the same. Facing a dropped connection a client SHOULD try to make zone determination whether the connection closure was the result of network activity or a decision by the AXFR server. This determination is not an exact science. It is up to the AXFR client implementor to react, but the reaction SHOULD NOT be an endless cycle of retries nor an increasing (in frequency) retry rate.

An AXFR server implementor SHOULD take into consideration the dilemma described above when a connection is closed with an outstanding query in the pipeline. For this reason, a server ought to reserve the course of action for situations in which it believes beyond a doubt that the AXFR client is attempting abusive behavior.

3 Zone Contents

The objective of the AXFR session is to request and transfer the contents of a zone. The objective is to permit the client to reconstruct the zone as it exists at the server for the given zone serial number. Over time the definition of a zone has evolved from a static set of records to a dynamically updated set of records to a continually regenerated set of records.

3.1 Records to Include

In the answer section of AXFR response messages the resource records within a zone for the given serial number MUST appear. The definition of what belongs in a zone is described in RFC 1034, Section 4.2, "How the database is divided into zones", and in particular, section 4.2.1, "Technical considerations".

Unless the AXFR server knows that the AXFR client expects just one resource record per AXFR response message, an AXFR server SHOULD populate an AXFR response message with as many complete resource records as will fit within a DNS message.

Zones for which it is impractical to list the entire zones for a serial number (Because changes happen too quickly) are not suitable for AXFR retrieval. A typical (but not limiting) description of such a zone is a zone consisting of responses generated via other database lookups and/or computed based upon ever changing data. In essence, if the zone changes (on average) more frequently than an AXFR session can be finished, the zone is not a good candidate for AXFR.

3.2 Delegation Records

In RFC 1034, section 4.2.1, this text appears (keep in mind that the "should" in the quotation predate [NS14], cf. section 1.1): "The RRs that describe cuts ... should be exactly the same as the corresponding RRs in the top node of the subzone." There has been some controversy over this statement and the impact on which NS resource records are included in a zone transfer.

The phrase "that describe cuts" is a reference to the NS set and applicable glue records. It does not mean that the cut points and the apex resource records are identical. For example, the SOA resource record is only found at the apex, as well as DNSSEC resource records. The is even a DNSSEC resource record found only at the zone cut and not at the corresponding apex. There are also some DNSSEC resource record sets that are explicitly different between the cut point and the apex. The discussion here is restricted to just the NS resource record set and glue as these "describe cuts."

The issue is that in operations there are times when the NS resource records for a zone might be different at a cut point in the parent and at the apex of a zone. Sometimes this is the result of an error and sometimes it is part of an ongoing change in name servers. The DNS protocol is robust enough to overcome inconsistencies up to (but not including) there being no parent indicated NS resource record referencing a server that is able to serve the child zone. This robustness is one quality that has fueled the success of the DNS. Still, the inconsistency is an error state and steps need to be taken to make it apparent (if it is unplanned) and to make it clear once the inconsistency has been removed.

Another issue is that the AXFR server could be authoritative for a different set of zones than the AXFR client. It is possible that the AXFR server be authoritative for both halves of an inconsistent cut point and that the AXFR client is authoritative for just the parent of the cut point.

The question that arises is, when facing a situation in which a cut point's NS resource records do not match the authoritative set, whether an AXFR server responds with the NS resource record set that is in the zone being transferred or is at the authoritative location.

The AXFR response MUST contain the cut point NS resource record set registered with the zone whether it agrees with the authoritative set or not. "Registered with" can be widely interpreted to include data
residing in the zone file of the zone for the particular serial number (in zone file environments) or as any data configured to be in the zone (database), statically or dynamically.

The reasons for this requirement are:

1) The AXFR server might not be able to determine that there is an inconsistency given local data, hence requiring consistency would mean a lot more needed work and even network retrieval of data. An authoritative server ought not be required to perform any queries.

2) By transferring the inconsistent NS resource records from a server that is authoritative for both the cut point and the apex to a client that is not authoritative for both, the error is exposed. For example, an authorized administrator can manually request the AXFR and inspect the results to see the inconsistent records. (A server authoritative for both halves would always answer from the more authoritative set, concealing the error.)

3) The inconsistent NS resource record set might indicate a problem in a registration database.

4) Beginning with an error state of two servers for a zone having inconsistent zone contents for a given zone serial number, if a client requests and receives an IXFR transfer from one server followed by another IXFR transfer from the other server, the client can encounter an IXFR protocol error state where an attempt is made to incrementally add a record that already exists or to delete a record that does not exist.

(Editorial note, the 4th reason was suggested, but I don’t see how it relaxes. A nudge for updated text on this.)

3.3 Glue Records

As quoted in the previous section, section 4.2.1 of RFC 1034 provides guidance and rationale for the inclusion of glue records as part of an AXFR transfer. And, as also argued in the previous section of this document, even when there is an inconsistency between the address in a glue record and the authoritative copy of the name server’s address, the glue resource record that is registered as part of the zone for that serial number is to be included.

This applies to glue records for any address family [RFC1700].

The AXFR response MUST contain the appropriate glue records as registered with the zone. The interpretation of “registered with” in the previous section applies here. Inconsistent glue records are an operational matter.

3.4 Name Compression

Compression of names in DNS messages is described in RFC 1035, section 4.1.4, "Message compression". The issue highlighted here relates to a comment made in RFC 1034, section 3.1, "Name space specifications and terminology" which says "When you receive a domain name or label, you should preserve its case." ("Should" in the quote predates [RFC14].)

Name compression in an AXFR message MUST preserve the case of the original domain name. That is, although when comparing a domain name, "a" equals "A", when comparing for the purposes of message compression, "a" is not equal to "A". Note that this is not the usual definition of name comparison in the DNS protocol and represents a new requirement on AXFR servers.

Rules governing name compression of RDATA in an AXFR message MUST abide by the specification in "Handling of Unknown DNS Resource Record (RR) Types" [RFC1597], specifically, section 4 on "Domain Name Compression."

3.5 Occluded Names

Dynamic Update [RFC2136] (and including DNAME [RFC2672]) operations can have a side effect of occluding names in a zone. The addition of a delegation point via dynamic update will render all subordinate domain names to be in a limbo, still part of the zone but not available to the lookup process. The addition of a DNAME resource record has the same impact. The subordinate names are said to be "occluded."

Occluded names MUST be included in AXFR responses. An AXFR client MUST be able to identify and handle occluded names. The rationale for this action is based on a speedy recovery if the dynamic update operation was in error and is to be undone.

4 Transport

AXFR sessions are currently restricted to TCP by section 4.3.5 of RFC 1034 that states: "Because accuracy is essential, TCP or some other reliable protocol must be used for AXFR requests." The restriction to TCP is also mentioned in section 6.1.3.2. of "Requirements for Internet Hosts - Application and Support" [RFC1231].

The most common scenario is for an AXFR client to open a TCP connection to the AXFR server, send an AXFR query, receive the AXFR response, and then close the connection. There are variations on this, such as an query for the zone’s SOA resource record first, and so on. Note that this is documented as a most common scenario.

The assumption that a TCP connection is dedicated to the single AXFR session is incorrect, this has lead to implementation choices that prevent either multiple concurrent zone transfers or the use of the open connection for other queries.

Being able to have multiple concurrent zone transfers is considered desirable by operators who have sets of name servers that are authoritative for a common set of zones. It would be desirable if the name server implementations did not have to wait for one zone to transfer before the next could begin. The desire here is to tighten the specification, not a change, but adding words to the unclear areas, to define what is needed to permit two servers to share a TCP connection among concurrent AXFR sessions. The challenge is to design this in a way that can fall back to the old behavior if either the AXFR client or AXFR server is incapable of performing multiple concurrent AXFR sessions.
With the addition of EDNS0 and applications which require many small zones such as in web hosting and some ENUM scenarios, AXFR sessions on UDP are now possible and desirable. However, there are still some aspects of the AXFR session that are not easily translated to UDP. This document leaves AXFR over UDP undefined.

4.1 TCP

In the original definition there is an implicit assumption (probably unintentional) that a TCP connection is used for one and only one AXFR session. This is evidenced in no requirement to copy neither the Query Section nor the message ID in responses, no explicit ordering information within the AXFR response messages and the lack of an explicit notice indicating that a zone transfer continues in the next message.

The guidance given here is intended to enable better performance of the AXFR exchange as well as guidelines on interactions with older software. Better performance includes being able to multiply DNS message exchanges including zone transfer sessions. Guidelines for interacting with older software are generally applicable to AXFR clients as reversing the situation, older AXFR client and newer AXFR server ought to induce the server to operate within the specific for an older server.

4.1.1 AXFR client TCP

An AXFR client MAY request an connection to an AXFR server for any reason. An AXFR client SHOULD close the connection when there is no apparent need to use the connection for some time period. The AXFR server ought not have to maintain idle connections, the burden of connection closure ought to be on the client. Apparent need for the connection is a judgment for the AXFR client and the DNS client. If the connection is used for multiple sessions, or if it is known sessions will be coming or if there is other query/response traffic anticipated or currently on the open connection, then there is "apparent need."

An AXFR client MAY cancel delivery of a zone only by closing the connection. However, this action will also cancel all other outstanding activity using the connection. There is no other mechanism by which an AXFR response can be cancelled.

When a TCP connection is closed remotely (relative to the client), whether by the AXFR server or due to a network event, the AXFR client MUST cancel all outstanding sessions. Recovery from this situation is not straightforward. If the disruption was a spurious event, attempting to restart the connection would be proper. If the disruption was caused by a medium or long term disruption, the AXFR client would be wise to not spend too many resources trying to rebuild the connection. Finally, if the connection was dropped because of a policy at the AXFR server (as is the case with older AXFR servers), the AXFR client would be wise to not retry the connection. Unfortunately, knowing which of the three cases above applies is not clear (momentary disruption, failure, policy).

An AXFR client MAY use an already opened TCP connection to start an AXFR session. Using an existing open connection is RECOMMENDED over opening a new connection. (Non-AXFR session traffic can also use an open connection.) If in doing so the AXFR client realizes that the responses cannot be properly differentiated (lack of matching query IDs for example) or the connection is terminated for a remote reason, then the AXFR client SHOULD NOT attempt to reuse an open connection. If in doing so the AXFR client realizes that the responses cannot be properly differentiated (lack of matching query IDs for example) or the connection is terminated for a remote reason, then the AXFR client SHOULD NOT attempt to reuse an open connection.

An AXFR client MAY use an already opened TCP connection to start an AXFR session. Using an existing open connection is RECOMMENDED over opening a new connection. (Non-AXFR session traffic can also use an open connection.) If in doing so the AXFR client realizes that the responses cannot be properly differentiated (lack of matching query IDs for example) or the connection is terminated for a remote reason, then the AXFR client SHOULD NOT attempt to reuse an open connection.

An AXFR client MAY cancel delivery of a zone only by closing the connection. However, this action will also cancel all other outstanding activity using the connection. There is no other mechanism by which an AXFR response can be cancelled.

When a TCP connection is closed remotely (relative to the client), whether by the AXFR server or due to a network event, the AXFR client MUST cancel all outstanding sessions. Recovery from this situation is not straightforward. If the disruption was a spurious event, attempting to restart the connection would be proper. If the disruption was caused by a medium or long term disruption, the AXFR client would be wise to not spend too many resources trying to rebuild the connection. Finally, if the connection was dropped because of a policy at the AXFR server (as is the case with older AXFR servers), the AXFR client would be wise to not retry the connection. Unfortunately, knowing which of the three cases above applies is not clear (momentary disruption, failure, policy).

4.1.2 AXFR server TCP

An AXFR server MUST be able to handle multiple AXFR sessions on a single TCP connection, as well as handle other query/response sessions.

If a TCP connection is closed remotely, the AXFR server MUST cancel all AXFR sessions in place. No retry activity is necessary, that is initiated by the AXFR client.

Local policy MAY dictate that a TCP connection is to be closed. Such as action SHOULD be in reaction to limits such as those placed on the number of outstanding open connections. Closing a connection in response to a suspected security event SHOULD be done only in extreme cases, when the server is certain the action is warranted. An isolated request for a zone not on the AXFR server SHOULD receive a response with the appropriate return code and not see the connection broken.

4.2 UDP

AXFR sessions over UDP transport are not defined.

5 Authorization

A zone administrator has the option to restrict AXFR access to a zone. This was not envisioned in the original design of the DNS but has emerged as a requirement as the DNS has evolved. Restrictions on AXFR could be for various reasons including a desire (or in some instances, having a legal requirement) to keep the bulk version of the zone concealed or to prevent the servers from handling the load incurred in serving AXFR.

A DNS implementation SHOULD provide means to restrict AXFR sessions to specific clients.

An implementation SHOULD allow access to be granted to Internet Protocol addresses and ranges, regardless of whether a source address could be spoofed. Combining this with techniques such as Virtual Private Networks (VPN) [RFC2246] and/or "DNS Request and Transaction Signatures (SIGs)" [RFC2845] and/or "DNS Request and Transaction Signatures (SIGs)"

A general purpose implementation is RECOMMENDED to implement access controls based upon "Secret Key Transaction Authentication for DNS" [RFC2845] and/or "DNS Request and Transaction Signatures (SIGs)".
A general purpose implementation SHOULD allow access to be open to all AXFR requests. I.e., an operator ought to be able to allow any AXFR query to be granted.

A general purpose implementation SHOULD NOT have a default policy for AXFR requests to be "open to all." For example, a default could be to restrict transfers to loopback address(es) and such.

6 Zone Integrity

An AXFR client MUST ensure that only a successfully transferred copy of the zone data can be used to serve this zone. Previous description and implementation practices have used a two-stage model of the whole zone synchronization procedure: Upon a trigger event (e.g., polling of SOA resource record detects change in the SOA serial number, or via DNS NOTIFY [RFC1996]), the AXFR session is initiated, whereby the zone data are saved in a zone file or data base (this latter step is necessary anyway to ensure proper restart of the server); upon successful completion of the AXFR operation and zone sanity checks, this data set is 'loaded' and made available for serving the zone in an atomic operation, and flagged 'valid' for use during the next restart of the DNS server; if any error is detected, this data set MUST be deleted, and the AXFR client MUST continue to serve the previous version of the zone, if it did before. The externally visible behavior of an AXFR client implementation MUST be equivalent to that of this two-stage model.

If a server rejects data contained in an AXFR session, the server SHOULD remember the serial number and not attempt to retrieve the same zone version again.

Ensuring that an AXFR client does not accept a forged copy of a zone is important to the security of a zone. If a zone operator has the opportunity, protection can be afforded via dedicated links, physical or virtual via a VPN among the authoritative servers. But there are instances in which zone operators have no choice but to run AXFR sessions over the global public Internet.

Section 4.3.5 of RFC 1034 contains the following paragraph:

"The periodic polling of the secondary servers is controlled by parameters in the SOA RR for the zone, which set the minimum acceptable polling intervals. The parameters are called REFRESH, RETRY, and EXPIRE. Whenever a new zone is loaded in a secondary, the secondary waits REFRESH seconds before checking with the primary for a new serial. If this check cannot be completed, new checks are started every RETRY seconds. The check is a simple query to the primary for the SOA RR of the zone. If the serial field in the secondary's zone copy is equal to the serial returned by the primary, then no changes have occurred, and the secondary finds it impossible to perform a serial check for the EXPIRE interval, it must assume that its copy of the zone is obsolete and discard it."

Perhaps what is not clear in the paragraph regarding the EXPIRE interval time is that it is only reset to the EXPIRE parameter when a new zone is loaded. A new zone means a zone with a higher serial number than the most recently loaded zone. The EXPIRE interval timer is not reset automatically as a result of a zone transfer as a zone could be (mistakenly) transferred with the same or lower serial number.

I.e., successively transferring a zone from server to server does not permit the zone to avoid expiration.

8 Backwards Compatibility

Describing backwards compatibility is difficult because of the lack of specifics in the original definition. In this section some hints at building in backwards compatibility are given, mostly repeated from the earlier sections.

Backwards compatibility is not necessary, but the greater extent of an implementation's compatibility increases it's interoperability. For turnkey implementations this is not usually a concern. For general purpose implementations this takes on varying levels of importance depending on the implementer's desire to maintain interoperability. It is unfortunate that a need to fall back to older behavior cannot be discovered, hence needs to be noted in a configuration file. An implementation SHOULD, in it's documentation, encourage operators to periodically review AXFR clients and servers it has made notes about as instances in which zone operators have no choice but to run AXFR sessions over the global public Internet.

8.1 Server

An AXFR server has the luxury of being able to react to an AXFR client's abilities with the exception of knowing if the client can accept multiple resource records per AXFR response message. The knowledge that a client is so restricted apparently cannot be discovered, hence it has to be set by configuration.

An implementation of an AXFR server SHOULD permit configuring, on a per AXFR client basis, a need to revert to single resource record per message. The default SHOULD be to use multiple records per message.

8.2 Client

An AXFR client has the opportunity to try extensions when querying an AXFR server.

Attempting to issue multiple DNS queries over a TCP transport for an AXFR session SHOULD be aborted if it interrupts the original request and SHOULD take into consideration whether the AXFR server intends to close the connection immediately upon completion of the original
Concerns regarding authorization, traffic flooding, and message integrity are mentioned in "Authorization" (section 5), "TCP" (section 4.2) and "Zone Integrity" (section 6).

10 IANA Considerations

No new registries or new registrations are included in this document.

11 Internationalization Considerations

It is assumed that supporting of international domain names has been solved via "Internationalizing Domain Names in Applications (IDNA)" [RFC3490].

12 Acknowledgements

Earlier editions of this document have been edited by Andreas Gustafsson. In his latest version, this acknowledgement appeared.

"Many people have contributed input and commentary to earlier versions of this document, including but not limited to Bob Halley, Dan Bernstein, Eric A. Hall, Josh Littlefield, Kevin Darcy, Robert Elz, Levon Esibov, Mark Andrews, Michael Patton, Peter Koch, Ian Todd, and Brian Wellington."

Comments since the -05 version have come from these individuals: Alfred Hynes, Mark Andrews, Paul Vixie, Nouter Wijngaards, Iain Calder, Tony Finch, Ian Jackson, Andreas Gustafsson, Brian Wellington, ...
Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).