IPv6 and App Review

61k

This post discusses some of the more common reasons for why your app might encounter networking problems during App Review.

Not All Rejections Are IPv6 Related

The App Review rejection you've received indicates that:

- Your app failed
- It was tested on an IPv6-only network

This does *not* mean that the failure was specifically caused by that network. There are many other reasons why an app can work in your office but fail during App Review. QA1764 How to reproduce bugs reported against App Store submissions outlines our standard steps for isolating problems like this.

Specifically, if your app crashes during App Review then that's clearly a bug in your app; your app shouldn't crash regardless of the state of the network.

App Review's IPv6-only Network

App Review tests your app on an IPv6-only network that supports DNS64/NAT64 (**ARNAT64**). You can test your app in a similar way using an Internet Sharing-based DNS64/NAT64 test network (**MacNAT64**). FAQ #1 of the Supporting IPv6-only Networks pinned post has links to the instructions for setting this up. You should make sure your app works on this test network before going any further.

MacNAT64 is not *exactly* the same as ARNAT64 but it should be very close. Later sections of this document discuss some of the potential pitfalls here.

Check All the Servers

When evaluating the IPv6 readiness of your app, it's important to consider all of the servers it contacts. This may not be easy to determine. Some apps address many different servers for many different reasons (core functionality, secondary functionality, advertising, analytics, and so on). Moreover, it may not be easy for you to uncover all of these servers. For example:

- Server names may be hidden within library code
- Server names may be returned to your app as the result of other network requests
- For HTTP and HTTPS servers, your app may start off talking to one server and be redirected to a different server
- DNS

  `CNAME`

  records mean that a server can have many aliases

One easy way to uncover all of the servers used by your app is to look at a CFNetwork diagnostic log:

1. Enable CFNetwork diagnostic logging for your app, per the instructions in QA1887 CFNetwork Diagnostic Logging
2. Exercise your app in the usual way
3. Grab the log and filter it to build a list of servers

IMPORTANT This approach works well if your app uses high-level networking APIs (anything based on CFSocketStream, including NSURLSession and NSURLConnection). It will not show any networking done using low-level networking APIs, like BSD Sockets. You will have to investigate such code manually.

The rest of this document assumes your app is talking to one server. If your app is talking to multiple servers, you'll have to repeat the following steps for each one.

IPv6 Connectivity

The Networking Overview says:

> Note that, unlike DNS64/NAT64 workflows deployed by service providers, [MacNAT64] always generates synthesized IPv6 addresses. Therefore, it does not [support IPv6 connections] to IPv6-only servers outside of your local network

Historically ARNAT64 worked like a service provider DNS64/NAT64 network (**CellNAT64**), in that it provided direct IPv6-to-IPv6 connectivity. This is no longer the case, and ARNAT64 works much more like MacNAT64.

This approach has both pros and cons:

- If your app is compatible with MacNAT64, it's very likely to work on ARNAT64
- If there is some subtle IPv6 incompatibility, it will only show up on CellNAT64

Examples of the latter include:

- The DNS name being incorrect
- The DNS is correct but the server is not listening on IPv6
- The server is listening on IPv6 but fails when a request comes in over IPv6

The following sections discuss these points in turn. If you encounter problems on CellNAT64 that don't show up on MacNAT64, you should read through these sections for hints as to what might be going wrong.

IPv6 DNS Problems

Your first step in debugging IPv6 connectivity should be to check that the DNS is returning reasonable results for your server's name. You can do this using the

`dig`

command line tool. The following is an example of the results you'd expect if your server supports IPv6.

```
$ dig +nocmd +nostats example.com AAAA
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24342
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com. IN AAAA

;; ANSWER SECTION:
example.com. 66159 IN AAAA 2606:2800:220:1:248:1893:25c8:1946
```

Note

`dig`

is a flexible tool for interrogating DNS name servers.

`dig`

does not use this system's DNS resolver, but rather talks to the DNS server directly. In these examples I'm supplying the

`+nocmd`

and

`+nostats`

arguments to minimise the noise in the output and

`AAAA`

to query for an IPv6 address record.
As you can see, the server responds with a status of

`NOERROR`

(line 3) and an Answer Section that contains the IPv6 address of the server. This indicates that the server claims to support IPv6.
In contrast, the following command shows the result to expect when the

`AAAA`

record is not present.

```
$ dig +nocmd +nostats ipv4only.arpa AAAA
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 49883
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;ipv4only.arpa. IN AAAA

;; AUTHORITY SECTION:
ipv4only.arpa. 489 IN SOA sns.dns.icann.org. noc.dns.icann.org. …
```

Note that the status is still

`NOERROR`

but there's no Answer Section. This indicates that the server only supports IPv4. If that's the case, you can skip the next few sections (move on to *General DNS Problems*) because your connections will always go through the NAT64 translator.
If

`dig`

is showing any other sort of response, read the next few sections to see if you've hit one of the more common DNS problems.

Common DNS Problem:

`SERVFAIL`

Status

One common DNS problem is illustrated by the example below:

```
$ dig +nocmd +nostats broken.example.com AAAA
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 27593
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;broken.example.com. IN AAAA
```

In this case the status (still on line 3) is

`SERVFAIL`

. This indicates a failure on the server rather than the absence of an

`AAAA`

record. If your DNS server incorrectly returns this status, it can cause problems for DNS64/NAT64 networks.
IMPORTANT Section 5.1.2 of RFC 6147 requires that a DNS64/NAT64 gateway treat a status of

`SERVFAIL`

when requesting an

`AAAA`

record as equivalent to

`NOERROR`

. However, it's possible that your app may be run on a DNS64/NAT64 network where the gateway does not handle this properly, so it's best if your DNS server not return

`SERVFAIL`

unless there's an actual failure.

Common DNS Problem:

`NXDOMAIN`

Status

Another common DNS problem is illustrated by this example:

```
$ dig +nocmd +nostats broken.example.com AAAA
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 52961
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;broken.example.com.        IN   AAAA
```

Again, the key thing to note here is the status (line 3). A status of

`NXDOMAIN`

means that there are *no records of any type for this name*. You shouldn't see this for your server's name because that server should have an

`A`

record containing its IPv4 address. Some DNS servers incorrectly return

`NXDOMAIN`

when asked for an

`AAAA`

record, even though there's a valid

`A`

record. A bug like this in your DNS server will cause problems for IPv6-only clients because the DNS64/NAT64 gateway takes the DNS server at its word: an

`NXDOMAIN`

result for the

`AAAA`

query means there is also no

`A`

record, and thus the name is invalid.

Common DNS Problem: IPv4-Mapped Addresses

Finally, you should watch out for responses like this:

```
$ dig +nocmd +nostats broken.example.com AAAA
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34278
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;example.com. IN AAAA

;; ANSWER SECTION:
example.com. 66159 IN AAAA ::ffff:93.184.216.34
```

The DNS returned an

`AAAA`

for the name but that record contains an IPv4-mapped address (these are shown as

`::ffff:x.x.x.x`

, where

`x.x.x.x`

is a standard IPv4 address). Such an address is not valid in an

`AAAA`

record and will cause problems for the DNS64/NAT64 gateway. You must reconfigure your server to either:

- Not claim to support IPv6, by removing the

  `AAAA`

  record
- Natively support IPv6, by putting a real IPv6 address in the

  `AAAA`

  record

IPv6 Listening

If your server claims to support IPv6, you should make sure it's listening on the IPv6 address returned by the DNS. If it's a web server (HTTP or HTTPS), there are a variety of web-based tools to check this (for example, ipv6-test.com). If it's some other sort of server, you will have to test this from a native IPv6 network.

IMPORTANT When testing a web server, make sure you know whether the server is being contacted by HTTP or HTTPS, and test the appropriate protocol.

IPv6 Server Failures

It's possible that your server is correctly set up to use IPv6 but fails when it gets a request over IPv6. If you can read the server's log, you might be able to use it to debug failures like this. If not, you will have to test your server from a native IPv6 network.

General DNS Problems

For DNS64/NAT64 to work properly, it's important that your DNS servers are set up correctly. It's easy for subtle DNS configuration errors to cause problems to show up in some circumstances and not in others, like on ARNAT64 but not MacNAT64, or CellNAT64 but not ARNAT64.

There are a variety of Internet-based tools you can use to check your DNS setup. Many of them are based on the DNSCheck code from The Internet Foundation In Sweden. While it's possible to download and run this code yourself, it's generally easier to use their online tool.

When checking DNS, be wary of the following:

- DNS aliases (

  `CNAME`

  records)
- Domain names versus zones

The following sections explain these points in detail.

DNS Aliases

As mentioned earlier, DNS allows a server to have multiple aliases via

`CNAME`

records. If the name you're connecting to is a DNS alias, and the source and destination names are in different zones, you have to check each zone independently.
You can uncover these DNS aliases using

`dig`

. For example, the following shows that

`ftp.microsoft.com`

is actually a

`CNAME`

for

`ftp.microsoft.akadns.net.`

, which in turn resolves to the IP address 134.170.188.232. On the other hand,

`www.example.com`

directly points to 93.184.216.34.

```
$ dig +short ftp.microsoft.com
ftp.microsoft.akadns.net.
134.170.188.232
$ dig +short www.example.com
93.184.216.34
```

In the case of

`ftp.microsoft.com`

, the DNS alias means you have to be concerned with both the

`microsoft.com.`

domain and the

`akadns.net.`

domain.

Domain Names versus Zones

Some DNS checking tools will not let you enter an arbitrary domain name. Instead, you have to enter the domain name of the top of the zone. For example, you can't check

`www.example.com`

, but instead have to check

`example.com`

.
You can find the top of the zone by searching for an

`SOA`

(Start of Authority) record. If you don't get an

`SOA`

record for the name, walk 'up' the DNS hierarchy, removing a label from the front of the name, until you do. The following example shows how to do this using

`dig`

.

```
$ dig +short www.example.com SOA
$ dig +short example.com SOA
sns.dns.icann.org. noc.dns.icann.org. …
```

Once you have the top of the zone you can check the domain using whatever DNS checking tool you choose. The following is an example of running a check with the

`dnscheck`

command line tool, which I built from the DNSCheck open source on my Mac.

```
$ /usr/local/bin/dnscheck example.com
  0.000: example.com INFO Begin testing zone example.com with version 1.6.6.
  0.001: example.com INFO Begin testing delegation for example.com.
 17.579: example.com INFO Name servers listed at parent: a.iana-servers.net,b.iana-servers.net
 22.148: example.com INFO Name servers listed at child: a.iana-servers.net,b.iana-servers.net
…
```

Note Installing

`dnscheck`

is not without its challenges. If you only have a few domains to check, it's *much* easier to use IIS's online tool.

Well-Known Prefix

Some developers have attempted to synthesise an IPv6 address by combining an IPv4 address with the Well-Known Prefix (64:ff9b::/96). This will not work in the general case. For reliable results you must use

`getaddrinfo`

(or some higher-level API) to synthesise IPv6 addresses, as described in Use System APIs to Synthesize IPv6 Addresses. For more advice on this front, see FAQ #4 in the Supporting IPv6-only Networks pinned post.

Testing With Cellular Carrier DNS64/NAT64

The ultimate acid test for DNS64/NAT64 compatibility is whether your app works on a CellNAT64 network. Many cellular carriers around the world now put iOS devices on their DNS64/NAT64 network. You should investigate whether such a carrier is available in your region and, if so, get a SIM from that carrier and test on their network.

When you run this test make sure to disable Wi-Fi so that your connections go over WWAN, and thence over CellNAT64.

Note If you're based in the US, or have folks who are based in the US who can help you out, you should test your app on T-Mobile. T-Mobile puts iOS devices on their CellNAT64 if they are running iOS 10.3 or later.

Share and Enjoy — Quinn "The Eskimo!" Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

Change History

- 23 Jun 2016 — First posted.
- 22 Jul 2016 — Removed an incorrect mention of

  `NXDOMAIN`

  and added a more in-depth discussion of that status. Expanded the discussion of

  `SERVFAIL`

  , adding a reference to Section 5.1.2 of RFC 6147. Made minor editoral changes.
- 23 Aug 2016 — The App Review DNS64/NAT64 gateway now handles

  `SERVFAIL`

  correctly; changed the discussion of that response code.
- 26 Sep 2016 — Specifically called out the issue with IPv4-mapped addresses in the DNS. Broke up the *IPv6 DNS Problems* section into subsections to make it easier to follow.
- 27 Sep 2017 — Added a note about crashing to the *Not All Rejections Are IPv6 Related* section. Updated *IPv6 Connectivity* to discuss changes made to ARNAT64 a while back. Added the *Testing With Cellular Carrier DNS64/NAT64* section. Other minor editorial changes.

Network

Asked 4 years ago by eskimo

Reply to this question