**<https://blog.cloudflare.com/introducing-cname-flattening-rfc-compliant-cnames-at-a-domains-root/>**

**Introducing CNAME Flattening: RFC-Compliant CNAMEs at a Domain's Root**

04/03/2014

[Matthew Prince](https://blog.cloudflare.com/author/matthew-prince)

8 min read

A close-up of a roller

Description automatically generated

This post is about a new feature we've been quietly rolling out over the last few months. Last week we began enabling it for everyone by default. It's called CNAME Flattening and it's a bit geeky, but very useful and important if you’re using cloud-based services and you hate having a “www” subdomain. The gist is: you can now safely use a [CNAME record](https://www.cloudflare.com/learning/dns/dns-records/dns-cname-record/), as opposed to an A record that points to a fixed IP address, as your root record in [CloudFlare DNS](https://www.cloudflare.com/dns/) without triggering a number of edge case error conditions because you’re violating the DNS spec. Geeky, I know, so before asking you to wade through why this is cool, here are some of our customers singing the feature’s praises:

A person driving a car with a stuffed animal

Description automatically generated

*"CloudFlare's CNAME Flattening feature enabled us to easily front our Amazon Elastic Load Balancer (ELB) with CloudFlare's DDoS and WAF protection."*

Gregory S. Woodward, Ph.D., CTO,[NASDAQ Private Market](https://www.nasdaqprivatemarket.com/)

*"This was a hard problem that needed solving. Leave it to the team at CloudFlare to come up with a great solution that will act as a gateway to a virtually unlimited number of other cloud services."*

Alex Polvi, CEO, [CoreOS](https://coreos.com/)

*"CNAME Flattening allowed us to use a root domain while still maintaining DNS fault-tolerance across multiple IP addresses."*

Andrew Warner, Director of Engineering, [RapGenius](http://genius.com/)

Intrigued? Read on to understand the technical details of exactly what it is and why we needed to build it.

**The Inflexibility of DNS**

Traditionally, the root record of a domain needed to point to an IP address (known as an A -- for "address" -- Record). While it may not seem like a big deal, tying a service to an IP address can be extremely limiting. Imagine that a new blogging platform, WordPlumblr, starts up. WordPlumblr allows its users to use custom domains that point to the WordPlumblr infrastructure. Foo.com signs up and WordPlumblr gives Foo.com an IP address. The supply of IP addresses is limited so, over time, as more sites sign up, IP addresses end up getting shared between multiple customers. No problem since with virtual hostnames WordPlumblr can return different content for different domains even hosted on the same IP address.

Everything is fine until Foo.com starts using too many of WordPlumblr's resources -- maybe because they're being attacked or they're featured on Oprah or who knows why. The other customers of WordPlumblr get poor performance because they're sharing the same resources as the overwhelmed Foo.com. WordPlumblr is put in a difficult position: having to reach out to Foo.com to get them to change their DNS settings or, even harder, having to reach out to all the other customers using the IP address to get them to change. Good luck.

**CNAMEs For the Win**

The solution is a CNAME. A CNAME is an alias. It allows one domain to point to another domain which, eventually if you follow the CNAME chain, will resolve to an A record and IP address. If WordPlumblr had handed out a unique CNAME for every customer then they wouldn't have had a problem. For example, WordPlumblr might have assigned the CNAME 6equj5.wordplumblr.com for Foo.com. Foo.com and the other customers may have all initially resolved, at the end of the CNAME chain, to the same IP address. However, when Foo.com started using too many resources WordPlumblr could have updated the CNAME and isolated Foo.com from the rest of the customers.

If you're familiar with programming, this is just like the idea of a pointer. When you're programming in C, in order to allow flexibility in memory management, you usually don't want to address memory directly but, instead, you set up a pointer to a block of memory where you're going to store something. If the operating system needs to move the memory around then it just updates the pointer to point to wherever the chunk of memory has been moved to. The program references the pointer and is none the wiser.

CNAMEs work great for subdomains like www.foo.com or blog.foo.com. Unfortunately, they don't work for a naked domain like foo.com itself. And, for reasons that somewhat perplex me, a lot of people are obsessed with using their naked domain for their website. So why don't CNAMEs work at the root?

**No Root For You**

A person pointing at a piece of paper

Description automatically generated

The problem stems from the fact that the [DNS specification](http://www.ietf.org/rfc/rfc1035.txt) dates from 1987. At the time, no one conceived of website -- it was two years before Tim Berners-Lee’s seminal paper first laying out the idea for the world wide web -- let alone modern outsourced cloud services like Amazon Web Services (AWS) or Heroku. As a result, the DNS spec enshrined that the root record -- the naked domain without any subdomain -- could not be a CNAME. Technically, the root could be a CNAME but the RFCs state that once a record has a CNAME it can't have any other entries associated with it: that's a problem for a root record like example.com because it will often have an MX record (so email gets delivered), an NS record (to find out which nameserver handles the zone) and an SOA record.

Because they follow this specification, most authoritative DNS servers won't allow you to include CNAME records at the root. At CloudFlare, [we decided to let our users include a CNAME at the root](https://blog.cloudflare.com/zone-apex-naked-domain-root-domain-cname-supp) even though we knew it violated the DNS specification. And that worked, most of the time. Unfortunately, there were a handful of edge cases that caused all sorts of problems.

**Break the RFC at Your Own Peril**

You'd never guess, but the biggest edge case had to do with email sent from Microsoft Exchange mail servers. Domains generally designate the servers that handle their email through what's known as a MX Record. The problem was that Exchange servers, under a very specific set of circumstances, could pick up the CNAME at the root record and then not properly respect the CNAME set at the MX record. You can't really blame Exchange. They were operating under the assumptions laid out by the DNS specification. However, this and a handful of other corner cases caused us to support but recommend against using a CNAME at the root record. Until now.

**Introducing CNAME Flattening**

A sign with green text

Description automatically generated

What we needed was a way to support a CNAME at the root, but still follow the RFC and return an IP address for any query for the root record. To accomplish this, we extended our authoritative DNS infrastructure to, in certain cases, act as a kind of DNS resolver. What happens is that, if there's a CNAME at the root, rather than returning that record directly we recurse through the CNAME chain ourselves until we find an A Record. At that point, we return the IP address associated with the A Record. This, effectively, "flattens" the CNAME chain.

This follows the DNS specification and is invisible to any service that interacts with our DNS. We've tested it extensively over the last several months and it works great, completely resolving the Microsoft Exchange and other edge case problems we'd previously seen.

**Flexible and Faster**

The biggest benefit is that this allows the flexibility of having CNAMEs at the root without breaking the DNS specification. An ancillary benefit we've found is that we decrease the time for CNAME resolution by about 30% on average. We cache the CNAME responses -- respecting the DNS TTLs, just like a recursor should -- which means often we have the answer without having to traverse the chain. When we do need to traverse the CNAME chain, we often have much a much faster, more direct connection to whatever server is authoritative than your visitor's ISP's recursive DNS service, which would have otherwise been doing the heavy lifting.

Cartoon a cartoon of a child waving

Description automatically generated

While we're defaulting this on just for root records, we've begun experimenting with turning CNAME Flattening on more broadly. We've begun to flatten all the CNAMEs that CloudFlare uses internally. For example, if you signed up via a hosting partner then you may have gotten a CNAME that included in the CNAME chain something like cf-ssl2463-protected-www.cloudflare.com.cdn.cloudflare.net. That is increasingly going away and will now resolve directly from your domain to an IP address. This makes DNS resolution a bit faster and also better obscures the fact that you're using CloudFlare.

As of last week, CNAME Flattening is on by default. To take advantage of it, just [add a CNAME to your root record](https://dash.cloudflare.com/?to=/:account/:zone/dns). The new feature is included for free for everyone using CloudFlare's DNS. If you've been looking for a way to make your root record work on a hosted service like AWS, Heroku, or that hot new blogging platform WordPlumblr, look no further.

And, if you still need some convincing, here are some more quotes from happy beta testers:

*"CNAME flattening solved email resolution errors for us which was very key. It also enables us to point our app hosted on skyprepapp.com to Amazon ELB and resolve emails there too."*

* Arash Barkhodaee, Co-Founder and CTO, [SkyPrep.com](https://skyprep.com/)

*"We chose to drop the www in our domain name around 6 years ago. Whilst I understand there are plenty of technical reasons to not use these APEX domains in URLs, in reality I don't like the idea of technical issues preventing us doing what we think is right. Our customers are internet savvy, and the www legacy is quite simply not needed anymore. So we dropped it and have used Amazon's Route 53 to date to balance our econsultancy.com traffic to our load balancers. When we moved across to CloudFlare, we had no intention to lose the Apex domain, but without CNAME flattening I do not believe this would have been possible. So this feature is very welcome and eased the migration to CloudFlare."*

* Matthew O'Riordan, CTO, [eConsultancy.com](https://econsultancy.com/)

*"CloudFlare's CNAME flattening solved a specific issue we saw with email resolution on our domain. When we first configured our DNS records for Junction we took advantage of a CloudFlare feature to set a CNAME record for our zone apex. This helped us resolve our root -- jct.com -- to an AWS ELB which routed traffic to our webservers and handled SSL termination. This worked well for resolving our http traffic, but we saw some issues with sending domains on older infrastructure that couldn't pick up our MX records correctly (we specifically saw this from senders on older versions of Microsoft SMTP Server). We worked with CloudFlare to isolate the problem and transition our domain to use their CNAME flattening solution. This effectively translates our CNAME to A records and keeps the IP addresses in sync with what is associated with our load balancer. All previous email problems have been resolved since we made the switch three months ago. A huge thanks to the CloudFlare team for getting a solution in place so quickly and an incredible level of support throughout the process."*

* Dennis Coldwell, Head of Platform, [Junction Investments](https://jct.com/)

Not using Cloudflare yet? [Sign up free](https://dash.cloudflare.com/sign-up) or choose one of our feature rich [paid plans](https://www.cloudflare.com/plans/#overview).

We protect [entire corporate networks](https://www.cloudflare.com/network-services/), help customers build [Internet-scale applications efficiently](https://workers.cloudflare.com/), accelerate any [website or Internet application](https://www.cloudflare.com/performance/accelerate-internet-applications/), [ward off DDoS attacks](https://www.cloudflare.com/ddos/), keep [hackers at bay](https://www.cloudflare.com/application-security/), and can help you on [your journey to Zero Trust](https://www.cloudflare.com/products/zero-trust/).  
  
Visit [1.1.1.1](https://one.one.one.one/) from any device to get started with our free app that makes your Internet faster and safer.  
  
To learn more about our mission to help build a better Internet, [start here](https://www.cloudflare.com/learning/what-is-cloudflare/). If you're looking for a new career direction, check out [our open positions](http://www.cloudflare.com/careers).