Address Resolution, Networking, and Cfdocument

Posted At : November 24, 2009 12:00 PM | Posted By : Mark Kruger
Related Categories: Hosting and Networking, Coldfusion Troubleshooting

Among the things that can befuddle even experienced developers, domain resolution ranks up at the top. Usually this is because we don't spend a lot of time worrying about resolution on our desktop or laptop or Iphone. DNS is an extremely mature technology and for the most part it just works with few issues. When it comes to a server however, there are several things that can trip up resolution. Without an understanding of exactly what is going on under the hood, you will find yourself destroying yet another keyboard with the ball of your fist as you shout "why won't you just work!"

Domain resolution comes into play on most ColdFusion applications, even if you don't think so. Among other things, resolution is important for:

- Data Sources - how do you connect to an external server?
- Ecommerce - how do you connect to a Gateway?
- Web Services - how do you create your stub class?

So let's take a short journey down this path and see if we can uncover some of the general principles that will help us troubleshoot domain resolution issues.

What Is It

Now some of my readers might be scratching their heads already. Perhaps you simply don't know what in the ham sandwich I'm talking about. So before we talk about how to troubleshoot resolution issues, let's take a moment and discuss exactly what domain resolution is all about. To start with it bears remembering that computers, for all their utility, don't actually handle "language" - at least not in the sense that you and I think of language. To be honest, computers aren't actually very good with language. That's why when you say "Handled a task" into your text recognition software it comes out "Handangled a trisket". What computers *are* really good at is handling numbers.

Every location on the internet can be addressed via a *number*. The number is actually 16 "bits" long (4 bytes, 16 0's or 1's in a row), but we have our own "friendly" way of looking at it. We use an "IP address" of 4 numbers between 0 and 254 separated by periods. Since we use friendly names like "coldfusionmuse.com" and the internet only uses numbers like 207.55.55.10, what's the key to figuring out which names match which numbers? Enter the "Domain Name Services" or DNS. When your browser makes a request for "coldfusionmuse.com" it accesses the "IP stack" (the network protocol software running on your computer) which has a conversation with a DNS server. All of this happens *before* your HTTP request is ever made.

- **Stack:** Yo.. DNS, I have a new one heh.
- **DNS:** All right give it to me Bob... and lose the accent. You're an Apple for goodness sake.
- **Stack:** Sorry, my user's been watching Goodfellas. Anyhow, he wants "coldfusionmuse.com". Can you give me the number for that one?
- **DNS:** Hang on... um... not in the cache. Hmmm... that domain is so immensely popular it is usually in the Cache. I'll have to make an outgoing query. [Hollering at Upstream DNS] hey Frank, you got a number for "coldfusionmuse.com"?
- **Frank:** Yes, I have that one. It's 207.55.55.10.
- **DNS:** Bob, the number is 207.55.55.10.

- **Stack:** Thanks... talk to you in a few milliseconds.
- **DNS:**Yeah yeah...

Using the number retrieved from DNS your browser then constructs an HTTP request and sends it to that IP address using the HTTP protocol. This process of figuring out the IP address of a given domain name is known as domain resolution. Actually you might hear the terms "IP address resolution" or "domain name resolution" cast about as well - and *most* of the time, unless they are talking about ARP, they are referring to the type of DNS conversation seen above.

Why Does Your Web Server Need It?

Web servers mostly *receive incoming traffic* from HTTP requests right? Why do they need to worry about address resolution? There are 3 reasons. 1) Web servers need to make outgoing requests and 2) requested resources are sometimes local to the web server and 3) most networks where web servers live are *not like the network you are accustomed to*.

Servers Make Outgoing Requests

Very often your server will make outgoing HTTP requests. Sometimes these requests are obvious. For example, when you using CFHTTP to a domain your server will first need to *resolve* the IP address of that domain, then make an outgoing request to that IP address. Your server functions exactly like a browser would, except your server will not be rendering the content.

Sometimes your server is making requests that are not obvious. One excellent example of this is when you use the CFDOCUMENT tag. In the rest of this post we will use Cfdocument to illustrate our points on resolution.

Our Fancy Pants Example

Take the following simple HTML:

```
<cfsavecontent variable="doc">
<div style="width: 350px; border: 1px solid #DDDDDD; padding: 10px;">

<div align="center">
<h2 align="center">Famous Authors</h2>
<img src="/images/dickens.jpg"/>
<h4>Charles Dickens</h4>
</div>
</div>
</cfsavecontent>
    <Cfoutput>#doc#</CFOUTPUT>
```

Pretty simple right? When your browser renders this code it looks up www.Whatever-Domain-You-Are-On.com and retrieves the file from the /images/ directory. By the way, when you are doing Cfdocument it is a good habit to use the method above. Save your content into a variable - that way you can test it as HTML and save yourself headaches trying to figure out why it doesn't render properly. To turn the above code into a Cfdocument all we would need is to wrap our cfoutputs in the cfdocument tag like so:

```
<Cfdocument format="PDF">
<Cfoutput>#doc#</CFOUTPUT>
```

```
</CFDOCUMENT>
```

So what happens when we render the cfdocument above? Instead of your browser, the ColdFusion engine must first figure out how to get that image. To do this, it is going to use (in this case) HTTP. That's right. Your server knows from the host header what domain is being used to call the page. It uses this domain internally to find the file. Just like your browser has to "call" for this file with an HTTP request separate from the main page, your CF Server must also call for this image in the same way. This is one reason why rendering pages as PDF files that have even a modest number of images or CSS files takes longer than may seem reasonable. A modest PDF with 5 images and a CSS file is going to make 6 outgoing HTTP requests, bring back the content and place it appropriately in the final version of the file. So you can see why domain resolution is important. Which brings us to our second point.

Resources Are Often Local

It may have slipped by you but the sample code above causes the CF server to call for a resource that resides *on itself*. This is pretty much the case with most PDF files. They most often are calling for resources that reside right within the same server, sometimes within the same folder as the code. I'm going to finish this discussion about domain resolution, but if you want a cool tip on how to avoid domain resolution for cfdocument altogether, check out my previous post on **Using the File System with Cfdocument**. Meanwhile, our cfdocument code is going to have to resolve your domain. And that leads us to our final and most important point.

Your Server's Network is not like Your Network

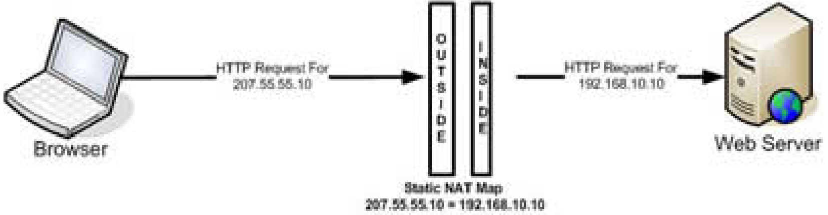
Now let me say at the beginning that there are millions of different ways to configure a hosting network. I'm only going to speak about one of them. The one I'm speaking about (and its many variations) is a pretty common way of setting up servers in a hosted network. If you have a better way or you like a particular product, bully for you. This post is about domain resolution and how this particular setup affects it. So please hold down the pitching to a dull roar. Ahem... let's proceed.

A hosted server is usually positioned behind a firewall (If it's not you should probably hang up your networking spurs right now). One of the more common ways of using a firewall is with what is often called a "static mapping" or sometimes a "One to one NAT address". The way it works is that the firewall exposes true internet addresses to the outside world. It *listens* for certain kinds of authorized traffic on these addresses through open listening sockets called "ports". So, for example, standard HTTP traffic usually comes through "port 80" and SSL traffic through "port 443". A web server usually has these two ports open and (hopefully) little else. These exposed public addresses are often thought of as the firewall's "outside" interface - the part facing "out" to the world.

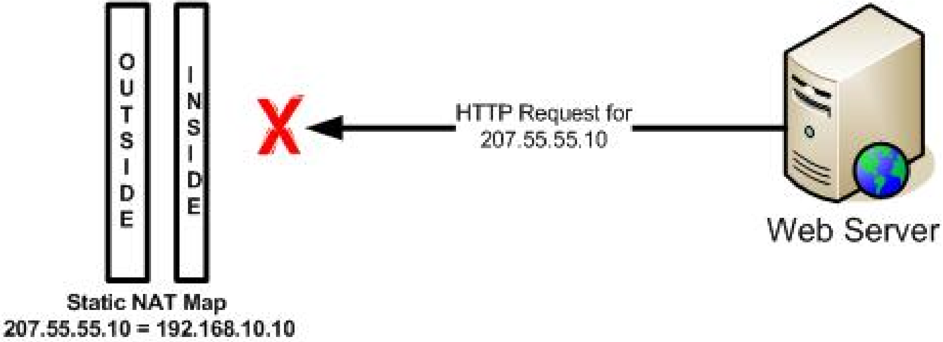
Now the firewall also has an "inside" interface facing the network. This interface is exposed to the web server, database server, and whatever other machines live inside the network and which access the outside world through the firewall. The *inside* addresses are not usually *public IP addresses*. Public IPs are pretty valuable these days and they are reserved for servers that need to be exposed to the world at large (like the web server on port 80). Internally the addresses are often what are called *non routable* IP addresses (they usually look like 192.x.x. or 10.x.x).

To allow traffic from the "outside" to the "inside" the firewall maintains a "Network

Address Translation" (NAT) table. Like a traffic cop it keeps track of what packets go where and supervises their delivery to the next hop. Servers are a little bit special inside of a network. There is often a 1 to 1 relationship between the outside address and the inside address. When this is the case we say the server IP is "statically mapped" or has a "one to one NAT address". For example, we could set up a rule that says, "207.55.55.10 = 192.168.10.10" - meaning any traffic hitting the *outside* address for IP 207.55.55.10 should be targeted *inside* for 192.168.10.10 (with all our filtering rules applied of course).



Let's keep that outside-inside image in focus as we go back to DNS and your CF server. When your ColdFusion server is rendering your Cfdocument and needs access to local resources (like an image on the same server and same domain) it queries DNS to get back an IP address. Here's the big question. Does the DNS server return the *outside* address or does it return the *inside* address. If DNS returns the *outside* address then your server will attempt to call that address to retrieve the image. But the server only has access to the *inside* interface. Many network firewalls (and load balancers) do not allow or are not capable of routing packets *inside to outside and back to inside*. Cisco Pix firewalls for example will not route this way. Instead your request is going to hang looking for a response from the *outside* interface that will never come.



How Do I Know if I Have This Problem

The easiest way to know is to SSH or RDP into the server and use a command line to try and resolve the address (ping or dig for example). Take a look at the server IP. If it's a non-routable IP and the DNS server returns a *public* IP then you *may* have this problem. You can also pull up a browser and simply browse to the resource that is timing out - but make sure you pull up the browser *on the server in question*. Browsing to it from your own workstation will tell you little if anything.

How Do I Solve It

Solving this issue depends on your level of control over the environment, but luckily you have several choices.

- **Internal DNS** - You can run a separate "internal" DNS server that serves internal IPs for servers inside the network. This works really well, but it does mean that DNS changes have to be done to both the external and Internal DNS servers.
- **HOSTS File Entry** - You can add an entry for the internal IP to your HOSTS file. This is quick and easy, but make sure and document it and don't forget it is there

or when the environment changes you will end up scratching your head trying to figure out why your domain is still resolving to the old IP.
- **File Method** - You can bypass HTTP and use the file method for local resources. Check out my previous post on **Using the File System with Cfdocument**. This is actually a neat tip. It will improve performance as well. But if your environment changes (file paths move for example) you run the risk of blowing this code up. Of course using expandPath() and some careful architecting can alleviate that as well. I would also recommend using locking since this method will involve file I/O.

Finally, let me say troubleshooting resolution should be systematic. Start by determining *can the server resolve the domain* then move to *is the resolved IP the correct IP* and so on. Systematic troubleshooting will yield the fastest result. Don't fall into the trap of trying your CFDOCUMENT over and over with different ways of doing it. If it is timing out, chances are you have resolution issues. Trust your instincts (or trust the Muse's if yours aren't well developed yet).