Troubleshooting CFHTTP - 3 Tips

Posted At : October 9, 2008 11:52 AM | Posted By : Mark Kruger
Related Categories: Coldfusion Troubleshooting

Hang around ColdFusion long enough and you will eventually find yourself making use of CFHTTP. The tag allows you to make an HTTP request from your Coldfusion server to an HTTP resource. Why would you want to do this? HTTP is a great low level protocol for data exchange. There's a reason why most web services run over HTTP - it is easy to understand and implement. As for CFHTTP, there are a great many examples, but here are a few.

- RSS Feeds - You might want to consume several RSS feeds and create a single feed from them. This is essentially what is going on at **Full As a Goog** and other sites like it.
- Ecommerce - Many Ecommerce gateways receive HTTP requests over SSL from web servers in order to process Credit card transactions.
- Data Files - You might use it to pick up text files for import or export.

Here's another example from the simple and practical side of my personal life. I listen to XM radio and I'm a baseball fan. The list of games and channels is online in the XM Radio **program guide**, but I wanted to access it from my mobile phone. I wanted to figure out my desired channel in the car without needing to scroll through dozens of channels. Using CFHTTP I unpacked the XM program guide for Major League Baseball, found the "print" version for the games, and figured out how to pass it today's date so that it would show the channels for *today's* game at the top. I created a server side script that adjusted the date and used CFHTTP to retrieve the file starting with today. The result can be viewed **here**. It loads fast on my Moto Q and now I can find the Cubs at a glance. Here are the 3 or 4 lines of code I needed to create the page.

```
<Cfset dt = dateformat(now(),'mm/dd/yyyy')/>

<cfset u =
"http://www.xmradio.com/schedule/sport/get_mlb_schedule.jsp?print=yes&startDate=#dt#"/>

<cfhttp url="#u#"/>

<Cfoutput>
    #cfhttp.filecontent#
</CFOUTPUT>
```

This simple example only scratches the surface of what is possible with CFHTTP. There are, however, some nuances to using it. Troubleshooting CFHTTP is challenging at times. Without the benefit of experience you can find yourself going bald in a hurry. So, here are 3 tips from the muse to help you troubleshoot CFHTTP.

DNS Issues and Connection Issues

The most common problem when using CFHTTP is simply that the server cannot resolve the URL or is blocked from access. When the problem is DNS you will get a short *"connection failure"* message in place of the cfhttp.filecontent variable. If the problem is that the URL is being resolved (DNS is working) but the content is being *blocked* from retrieval you might get a hanging request and a timeout (or you might get a response

from the blocking entity).

To troubleshoot this issue you need access to the server - remote access is fine, but you need to be able to issue HTTP requests *from the server*. It does you little good to load the URL into the browser of your workstation and say "see... it resolves." That doesn't tell you wether the server can resolve it or not. Remember, servers live in a very a different environment from your laptop or desktop. They are usually surrounded by devices, routing and rules intended to protect them. They also sometimes use a different DNS server (or at least configured differently) on the "inside" network where they reside.

So you will need to verify that the *server* can resolve your URL. That's step 1. If you determine that that the server cannot resolve the URL then you have a few choices. You can attempt to get it to resolve by configuring DNS, changing the network config etc. Or you can add a hosts file entry to circumvent DNS (this will not solve a problem where the request is blocked but it will get you past DNS resolution issues). Either way, you have to enable the server to "get at" the content.

SSL Issues

Using HTTP over SSL requires that the public and private keys work together to "handshake" and allow the request to succeed. The public keys have a hierarchy of "trust" that is based on the entities that issued the certs (like Verisign for example). The browser handles all of this internally and generally trusts about everyone (or at least asks if you *want* to trust a cert). But remember, you are not using a browser. You are using the JVM to make your request. Just like the browser, the JVM has a list of entities that it trusts as well. It's call the "trusted keystore". If your using CFHTTP to make SSL requests and having trouble, part of the reason could be because the cert is not yet "trusted" by the keystore. The fix is a cryptic procedure using a command line tool that ships with Java. The instructions may be found **here**. Note, troubleshooting this issue will require a bit more patience than just a connection issue, and if your cert is a "self-signed" cert you are going to need some Advil.

Compression

Actually, I just learned about this issue from my good friend and CF/SQL guru Mike Klostermeyer (who if he ever decided to blog could put me to shame). It turns out that if the request is made against an IIS server and the IIS server has compression enabled for serving web pages, the CFHTTP process might have trouble unpacking the results into something readable. The fix, as it turns out, is to add some headers as cfhttpparams - like so:

```
<cfset u = 'http://www.myexample.com'/>

    <cfhttp url="#u#" method="get">
        <cfhttpparam type="Header" name="Accept-Encoding" value="deflate;q=0">
        <cfhttpparam type="Header" name="TE" value="deflate;q=0">
    </cfhttp>
```

As I understand it, these headers instruct IIS to send the content uncompressed. If you ever find yourself tied in knots over this particular issue, this will keep you sane (or at least keep you from jumping out of a 4 story window).

If you are a muse reader and you have tips on troubleshooting CFHTTP, feel free to add to the discussion.