Branch Target Offset Error

Posted At : September 28, 2007 2:06 PM | Posted By : Mark Kruger
Related Categories: Coldfusion MX 7, Coldfusion Troubleshooting

Recently I got an error I had never seen before. That's news in itself. This error was especially cryptic. The template in question made a *CreateObject()* call to a CFC that was quite lengthy. In fact, this CFC was over 2000 lines of code (Coldfusion 7). When it tried to compile the code it threw an error that said *branch target offset too large for short*. You know you are a true geek when seeing an error you have never seen before raises your pulse. I started hunting for a solution - and I found one.

The Details I Could Gather

Here's my take. Please note, much of this is speculation. If you are going to correct something by posting a comment I appreciate it - but be kind. The muse has a thin skin when it comes to condescension. I believe this error comes from the compiler. In this case I think it comes from how the byte code compiler is arranged by Coldfusion. You may remember that Coldfusion is really a sort of template engine for Java byte code. A Coldfusion template is compiled down into java classes. So there is an intermediate interpreter that creates compilable Java code out of your Coldfusion, then it runs it through the compiler. I believe this error comes from the compiler and not from the interpreter - because I found reference to it elsewhere in Java projects.

The error is generated from the process of *branching*. The execution of instructions "branch" based on conditions. In the JVM this is sort of a "go to" road map. If some condition is met or not met the execution is handed to A or B respectively. I'm not sure what A or B represent - perhaps a pointer or a thread or whatever - but I do now that it is stored and calculated as a short. Perhaps some of my extra-geeky readers can shed some light on those technical details. In any case the problem appears to be that the interpreter generates locations or threads or pointers that are out of bounds for the data type short. In other words the interpreter generates a branch location greater than 32767 (the limitation of a short). When the code tries to compile the instructions for that branch it blows up saying "branch target offset too large for short".

The Culprit

The docs and posts I read all said to try various things to shorten the number of branches in a given class. Most of them suggested using an include. I did not like that idea in a CFC so I first tried creating a base component containing many of functions in the CFC (which was actually a collection of data entry functions and not much more). I moved about a third of the CFC into a base CFC and then extended the original CFC using this new one. This did not appear to work - the CFC still errored out.

next, I left the first 7 or 8 functions intact (they were simple) and removed the more complex functions setting them aside. I ran the *createObject()* code and the CFC compiled correctly. I added one function from the set aside functions and ran it again. Again it compiled correctly. I kept doing this and continued to be successful until I got to the very last function. When I added this last function in my error returned. This code contained queries designed to work against both Oracle and MSSQL. The function looked like this.

```
<cffunction ...>
<!--- arguments.. --->
```

```
<!--- ..setup --->
<cftransaction>
<cfquery name="rid" datasource="mydatsource">
    DELETE FROM myTable
    WHERE     myPK = #somevar#
</cfquery>
<cfswitch expression="#Dstype#">
    <cfcase value="MSSQL">
        ... a lengthy insert query
        designed for T-SQL
    </cfcase>
    <Cfcase value="Oracle">
        ... a lengthy insert query
        designed for PL/SQL.
    </CFCASE>

</cfswitch>
</cftransaction>
</cffunction>
```

I thought it about it and the first thing I tried was removing the *cftransaction*. Viola! This worked and the code compiled. It seems that wrapping a Switch/Case in a cftransaction results in lengthy branching. Indeed, when I moved the function into its own component and ran it I was able to generate the same error - signifying that the error had nothing to do with the length of my CFC. Since the original programmer intended to use Cftransaction I now had to solve the problem of getting it back into the code and getting the component to successfully compile.

The Fix

I could see what he was after. A single delete statement that would work against both target platforms and an insert statement designed for each of them. By wrapping the whole thing in a Cftransaction he was trying to ensure a roll-back on the delete if the insert failed. So I tried this:

```
<cffunction ...>
<!--- arguments.. --->
<!--- ..setup --->

<cfswitch expression="#Dstype#">
    <cfcase value="MSSQL">
    <cftransaction>
    <cfquery name="rid" datasource="mydatsource">
        DELETE FROM myTable
        WHERE     myPK = #somevar#
    </cfquery>
        ... a lengthy insert query
        designed for T-SQL
    </cftransaction>
    </cfcase>
    <cfcase value="Oracle">
    <cftransaction>
    <cfquery name="rid" datasource="mydatsource">
        DELETE FROM myTable
        WHERE     myPK = #somevar#
    </cfquery>
        ... a lengthy insert query
        designed for PL/SQL
    </cftransaction>
    </cfcase>
</cfswitch>

</cffunction>
```

I moved the delete query and the Cftransaction inside the case statement. Even though this resulted in several *more* lines of code added to the CFC it solved our branching issue. Cftransaction has to create additional roll-back branches. I think that adding a switch/case into the mix automatically creates a broad offset because it doubles the amount of work that the cftransaction will need to do to accomplish it's mission. Or something like that....