Source-less Deployment of ColdFusion Applications

When deploying applications to any 3rd party hosting provider or client, you may wish to consider protecting your development effort and intellectual properties (IP) by either encrypting or pre-compiling the application code.

In this document, I will briefly cover a number of ways in which you can accomplish this.

There are currently three methods of deploying your ColdFusion applications in an encrypted format, which ship with ColdFusion, so as to help protect your code:

1. Encrypting templates using the "cfencode" utility.
2. Pre-compiling and deploying templates as Java Byte code using the "cfcompile" utility.
3. Deployment Packages as J2EE Archives (EAR or WAR files) – accomplished using the ColdFusion Administrator.

Another option is packaging your application in ColdFusion Archive Files (CAR), but this is not within the scope of this document as there is no compilation or encryption used with this method.

Further details can be found at:
http://livedocs.adobe.com/coldfusion/8/htmldocs/help.html?content=deploying_3.html

The above options all have their own behaviours, pros and cons. These are discussed in more detail below.

Template encryption using "cfencode"

Pre CFMX7, ColdFusion had a utility, "cfencode", which encrypted your CFML templates, which in turn could be distributed as part of an application. ColdFusion knew how to decrypt that template and also how to execute it.

However, this was not a good solution as people came up with a utility to decode this encrypted file, therefore removing the benefit of protecting your IP. (cfdecrypt - by Matt Chapman http://www.zmatt.net/)

Another disadvantage of this method was that template execution was slower due to the fact that ColdFusion had to decrypt the template before compiling and executing it.

This option has more or less been superseded by the "cfcompile" utitlity, and therefore is NOT recommended as it has not been updated in line with the ColdFusion product releases.

NB: This utility is also unavailable for OS X.

Pre-Compilation using “cfcompile”

From CFMX7 onwards, ColdFusion developed a source-less deployment utility, which would make decryption of original code templates almost impossible.

The “cfcompile” utility can be used to simply compile your ColdFusion source code into the resulting Java class files. A benefit of which is that initial load times are quicker.

Secondly, with the “-deploy” option, it also allows for pre-compiling CFML code (CFM, CFC, and CFR templates) to Java byte code, which can then be deployed without the original source templates.

This utility, which can be found in the *cf_root*/bin (for the server configuration) and in the *cf_webapp_root*/WEB-INF/cfusionmx7/bin or *cf_webapp_root*/WEB-INF/cfusion/bin (for the multi-server and J2EE configuration) directory, can be run on one template or an entire directory structure containing a complete CFML application.

In doing so, cfcompile has its advantages:

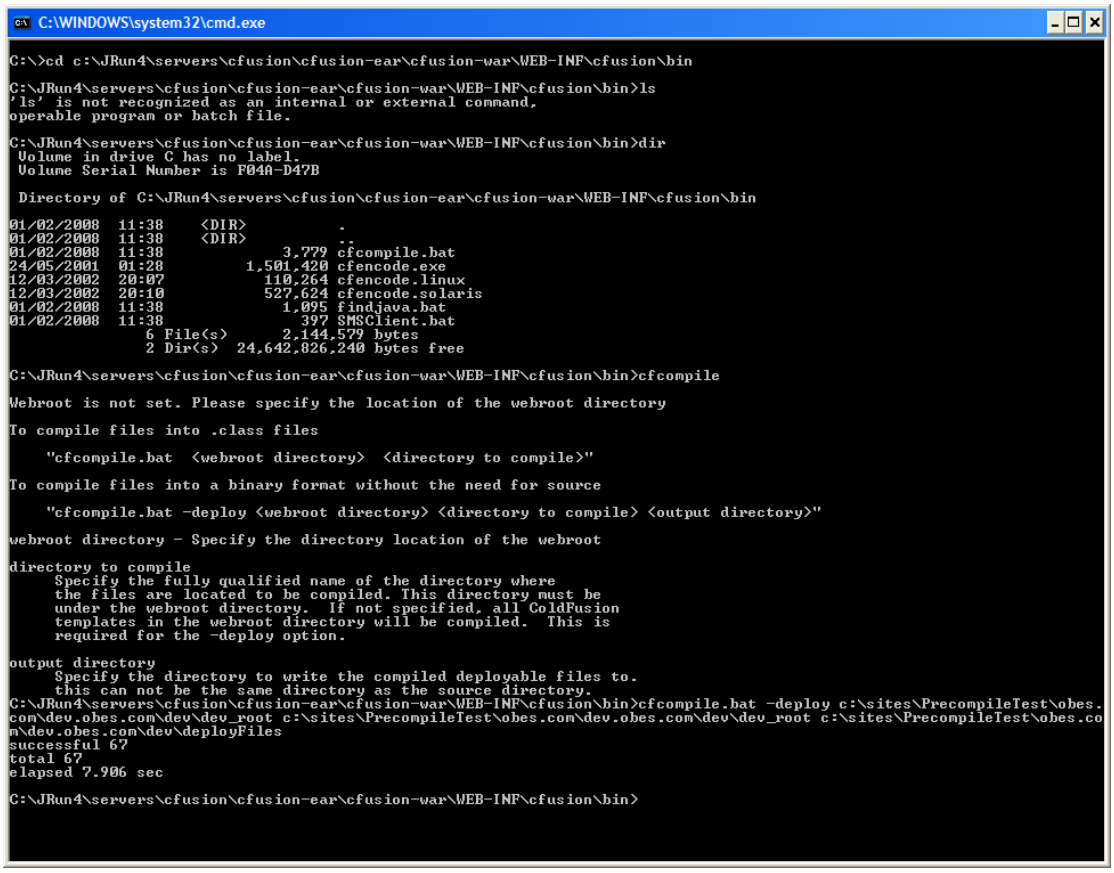
1. The code is encrypted and is not human readable, therefore offering some initial form of protection on your IP.
2. Because the code is pre-compiled – ColdFusion does not have to recompile the source.
3. Enhanced initial page load times at runtime.
4. The template retains its original name (i.e. login.cfm) when compiled.
5. Templates can be deployed to either Standard or Enterprise versions of ColdFusion server, so there are no extra licensing fees.
6. Can be integrated with other technologies for building and testing an application (see Summary).
7. Simple to use.

NOTE: Caution must be used when using the cfcompile utility. You must ensure that the original source is backed-up previously as this process is irreversible. Also, you can only deploy pre-compiled code to ColdFusion MX7+ Application servers. It is NOT backward compatible.

Using cfcompile is a manual process which only deals with the compilation of CFML templates into Java class files. For a complete deployment option, see the section title “WAR or EAR Distribution”.

Cfcompile Usage

Below is a screenshot of using the “cfcompile.bat” utility and it’s output.



Usage:

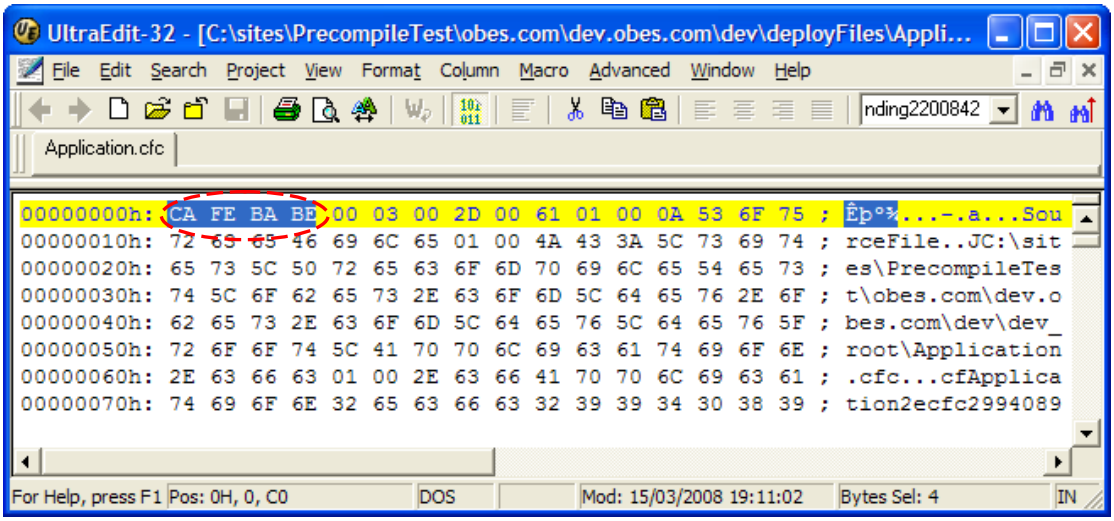
cfcompile –deploy web_root directory_to_compile output-directory

Parameter	Description
Web_root	Fully qualified path to the web server root
Directory-to-compile	Fully qualified path to the directory where the files to be compiled are located.
Output-directory	Fully qualified path to the directory to contain the compiled deployable files. This cannot be the same as the source directory!

Cfcompile Output Result

The below image is a screenshot of the Java byte code template generated by the cfcompile utility, opened in a hex editor. Clearly the code is encrypted and cannot be easily read.

Below, the highlighted hex characters represent the text “CAFEBABE”. This string should be found at the start of ALL CFML templates that have been compiled by the cfcompile utility with the “-deploy” option.



WAR or EAR Distribution

While the “cfcompile” utility satisfies the protection of the original source code, it still does not provide a complete deployment facility for ColdFusion applications.

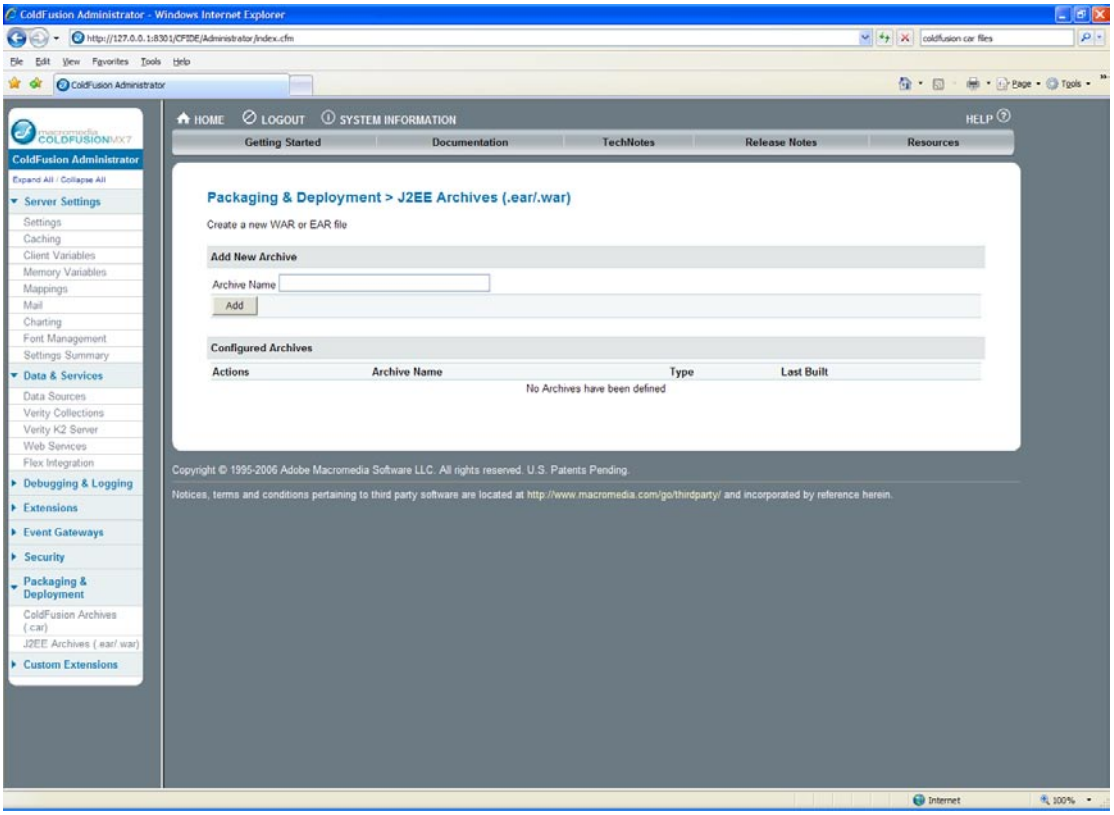
ColdFusion MX has been a pure J2EE application since its initial release.

A traditional J2EE application is bundled as an EAR (Enterprise Application Archive) or a WAR (Web Application Archive) file.

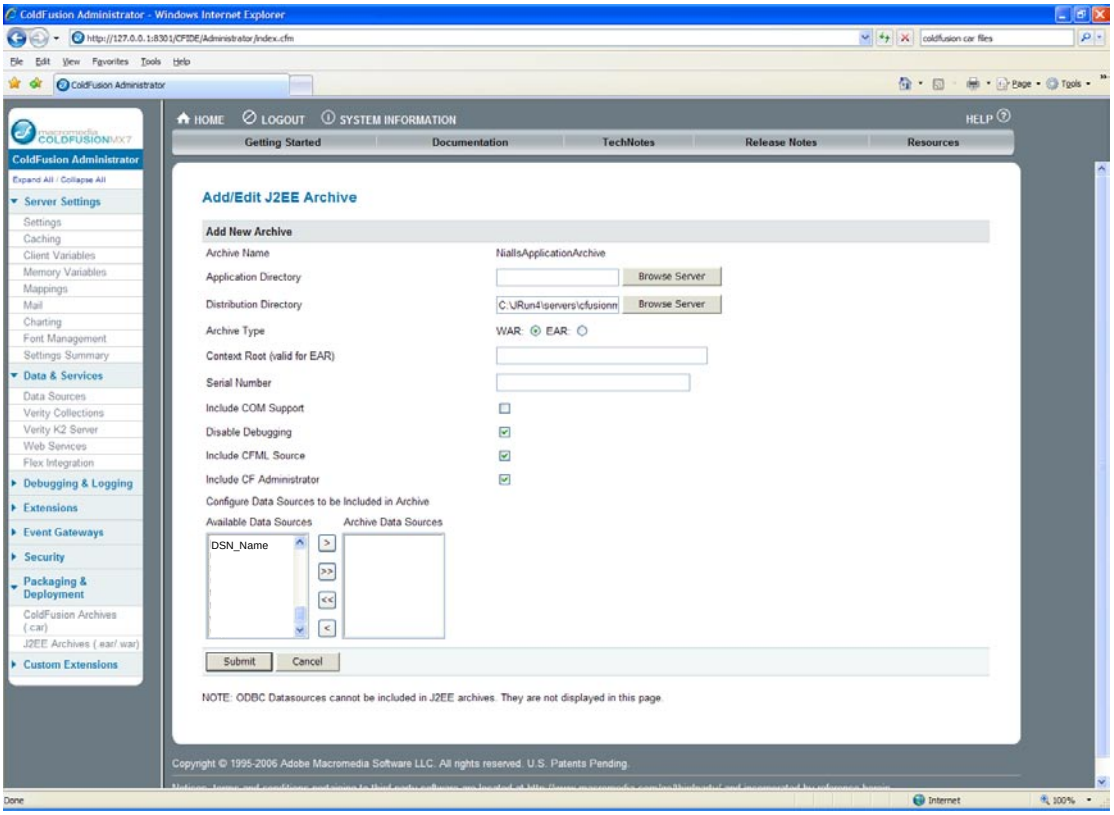
When running a J2EE configured environment, system administrators can simply take the EAR/WAR file and drop it into the J2EE instance. However, traditional ColdFusion deployments meant that a server first had to be installed (the ColdFusion runtime) and then the CFML source code installed separately.

To solve this, ColdFusion MX 7 introduced the option of packaging CFML applications as an EAR or WAR file so that one file can be handed off to the system administrator for deployment. This option means that deployment and management is easier for the system administrator.

Found under the "Packaging & Deployment" section in the ColdFusion Administrator (see below), the "J2EE Archives (.ear/.war)" option provides a variety of options such as whether or not to include the source code or whether or not to include the ColdFusion Administrator.



WAR or EAR Distribution (Continued)



Having created your EAR or WAR file, the resulting file can be deployed – with no separate ColdFusion server installation required!!

The only issue with this is that the client MUST have a ColdFusion Enterprise licence for the server that the package is deployed to. Plus this option is a little more difficult to implement than cfcompile, for example:

If a serial number is not applied when the EAR/WAR package is first created, then either the ColdFusion Administrator must be included with the package, or the Administrator API must be used so that the user can apply their license upon deployment.

If the Administrator is not included and no serial number is applied via the Administrator API, the application is deployed as a 30-day trial that reverts to the Developer Edition upon expiration (which is obviously restricted to “localhost” access plus 2 client machines).

Summary

In summary, ColdFusion offers a number of options for deploying our applications and I think that, based on the application and client requirements, either option could be chosen to suit the deployment of that particular application.

Also, the cfcompile utility could be utilized with other build \ deployment tools, such as Apache Ant in order to come up with a “more automated” development option, but this is out of the scope of this current document. For an overview of how this could work, please see:

http://www.coldfusiondeveloper.com.au/go/news/by/adam-howitts/20061107-use-eclipse-ant-and-cfcompilebat-to-precompile-your-code/