

---

# CryptoXpress™ CF

A ColdFusion® tag that supports “Strong” encryption and message digests.



## Installation & User Guide

Software Version	3.0
Document	CryptoXpressCF.doc
Published	08/15/2007

CFXWorks, Inc.  
5365 Chelsen Wood Drive, Duluth, Georgia 30097

Email: [sales@CFXWorks.com](mailto:sales@CFXWorks.com)

<http://www.CFXWorks.com>

Printed in the United States of America.

---

<b>1. INTRODUCTION.....</b>	<b>3</b>
1.1. WHAT IS CRYPTOXPRESS CF?.....	3
1.2. AES ENCRYPTION .....	3
1.3. TRIPLEDES (3DES) .....	4
1.4. MD5 MESSAGE DIGESTS.....	4
1.5. SHA1 MESSAGE DIGESTS .....	5
1.6. HEX ENCODING.....	5
1.7. BASE64 ENCODING.....	5
<b>2. CRYPTOXPRESS CF TAG PARAMETERS.....</b>	<b>5</b>
2.1. TAG PARAMETERS.....	5
2.2. COLDFUSION ENCRYPTION EXAMPLE: .....	7
2.3. COLDFUSION DECRYPTION EXAMPLE: .....	8
<b>3. CRYPTOXPRESS CF INSTALLATION.....</b>	<b>9</b>
3.1. DISTRIBUTION LIBRARY .....	9
3.2. SAMPLE COLDFUSION PROGRAMS.....	9
3.3. DOCUMENTATION.....	10
3.4. CFX_ENCRYPT_AES TAG COMPATIBILITY .....	10
3.5. .NET COMPATIBILITY .....	12
3.6. INSTALLATION ON COLDFUSION .....	12
3.7. MINIMUM SYSTEM REQUIREMENTS.....	15
3.8. EXPORT LIMITATIONS .....	15

---

# 1. INTRODUCTION

## 1.1. What is CryptoXpress CF?

**CryptoXpress CF** is a ColdFusion custom tag that encrypts and decrypts data using the AES or TripleDES (3DES) encryption algorithm. Specifically, this tag supports:

- 128-bit and 256-bit **AES encryption**
- **3DES encryption** (TripleDES )
- CBC mode of operation
- PKCS5Padding
- User provided encryption keys
- User provided initialization vector (salt)
- Migration from CFXWorks' CFX\_ENCRYPT\_AES ColdFusion tag.

**CryptoXpress CF** also supports digesting data using the following algorithm:

- MD5
- SHA1

Several ColdFusion demonstration programs are provided in the distribution zip file.

## 1.2. AES Encryption

This tag encrypts and decrypts text using the algorithm specified in the Federal Information Processing Standard (FIPS) for the Advanced Encryption Standard, [FIPS-197](#). This standard specifies Rijndael, sometimes referred to as AES, as the FIPS-approved symmetric encryption algorithm that should be used by U.S. Government organizations, agencies, and others to protect sensitive information.

AES is a block cipher (symmetric key) encryption algorithm that supports 128-bit, 192-bit and 256-bit key sizes. It was selected (10/02/2000) by the National Institute of Standards and Technology (NIST) as the new Federal Information Processing Standard (FIPS) for encryption. Effective 12/04/2001, Rijndael replaced DES as the FIPs standard. For a detailed discussion of Rijndael, and the Advanced Encryption Standard (AES) selection process, please visit web site <http://csrc.nist.gov/encryption/aes/> .

---

The FIPS standard also defines a certification suite to validate correct implementation of the algorithm. The tag has been validated using this suite.

The tag implements both the 128-bit and 256-bit key size. The result returned from the encryption routine is a hex representation of the binary encrypted data. The result returned from decryption is the original “clear” text.

### 1.3. TripleDES (3DES)

Some users require support for the Triple Data Encryption Algorithm (TripleDES or 3DES) as specified in [NIST Special Publication 800-67](#). 3DES is also supported by **CryptoXpress CF**. 3DES is still commonly used in some industries although it is thought to be considerably less secure than even AES 128-bit encryption,

### 1.4. MD5 Message Digests

This tag will also calculate a message digest using either MD5 or SHA1. The act of calculating a message digest is sometimes referred to as “digesting” the information.

A message digest (also sometimes referred to as a one-way hash function) is a fixed length computationally unique identifier corresponding to a set of data. The result of the algorithm is that each file or data string digested will map to a particular block of information called a message digest. The digest is not random; digesting the same unit of data with the same algorithm will always produce the same message digest.

MD5 belongs to a family of one-way hash functions called message digest algorithms. The MD5 system is defined in RFC 1321. MD5 takes a message of arbitrary length and produces as output a 128-bit message digest. It is conjectured that it is computationally infeasible to produce two different messages having the same message digest, or to produce any message having a given message digest.

RFC 1321 also defines a certification suite to validate correct implementation of the algorithm. **CryptoXpress CF** has been validated against this suite.

---

This tag digests either a data string or a file and creates a 16 byte message digest and returns it as a 32 byte HEX string.

## 1.5. SHA1 Message Digests

Some users prefer to use SHA1. SHA1 digests a string or a file and creates a 20 byte message digest and returns it as a 40 byte HEX string. The hex string is 40 bytes long. SHA1 is defined in RFC 3174. SHA1 tends to be the preferred over MD5 for Department of Defense related activities. It is believed to be more secure than MD5.

## 1.6. HEX Encoding

The tag will HEX encode and decode an ASCII string. HEX encoding is useful, especially when one wants to manipulate binary data. HEX encoding should not be considered as a method to secure information.

## 1.7. BASE64 Encoding

The tag will BASE64 encode and decode an ASCII string. BASE64 encoding is also useful, especially when one wants to manipulate binary data. BASE64 encoding should not be considered as a method to secure information.

# 2. CRYPTOXPRESS CF TAG PARAMETERS

## 2.1. Tag Parameters

<i>Parameter Name</i>	<i>Description</i>	<i>Comment</i>
CRYPTO_ACTION	Input Parameter: 1 - encrypt file 2 - decrypt file 5 - encrypt string 6 - decrypt string 9 - digest string 11 - digest file 30 - translate an ASCII string to	Default value is "1"  Encryption is performed using CBC mode with PKCS5Padding.

	<p><b>HEX</b></p> <p>31 – translate an ASCII string to BASE64</p> <p>32 – translate a HEX string to ASCII</p> <p>33 – translate a BASE64 string to ASCII</p>	
CRYPTO_OPTION	<p>Input Parameter:</p> <p>103 - AES128/PKCS5Padding/CBC</p> <p>104 – AES256/PKCS5Padding/CBC</p> <p>112 - TripleDES/PKCS5Padding/CBC</p> <p>401 – MD5</p> <p>402 – SHA1</p> <p>-----</p> <p>The following are provided to support compatibility with CFXWorks' original CFX_ENCRYPT_AES tag. See Section 3.4 of this document for details.</p> <p>901 – AES128/ECB</p> <p>902 – AES256/ECB</p> <p>903 – AES128/CBC</p> <p>904 – AES256/CBC</p>	
CRYPTO_IN	<p>Input Parameter:</p> <p>For action 5, 6 and 9 the value to be encrypted, decrypted or digested.</p> <p>For action 1, 2 and 11 the name of the input file.</p>	<p>Length must be greater than 0.</p> <p><b>For decryption this value must be a valid hex string.</b></p>
CRYPTO_OUT	<p>Output Parameter:</p> <p>For action 5, 6 and 9 the result value.</p> <p>For action 1, 2 and 11 the name of the output file.</p>	<p><b>For encryption or message digests the result will be represented as a hex encoded data string.</b></p> <p><b>For decryption the result will be represented in ASCII text.</b></p>
CRYPTO_KEY	<p>Input Parameter: User provided key submitted as an ASCII value.</p> <p>Note that CRYPTO_KEY takes precedence over CRYPTO_KEY_HEX.</p>	<p>Should be 16 characters long for AES 128-bit encryption, 32 characters long for AES 256-bit encryption and 24 characters long for 3DES. Short keys are accepted and right filled with 0x00.</p>
CRYPTO_KEY_HEX	<p>Input Parameter: User provided key submitted as a HEX value.</p>	<p>Should be 32 characters long for AES 128-bit encryption, 64 characters long for 256-bit AES encryption, and 48 characters long for 3DES. Short keys are accepted and right filled with 0x00.</p>
CRYPTO_IV	<p>Input Parameter:</p>	<p>Should be 16 characters long</p>

	User provided initialization vector (salt) submitted as an ASCII value.  Note that CRYPTO_KEY takes precedence over CRYPTO_KEY_HEX.	for AES and 8 characters long for 3DES. Short IVs are accepted and right filled with 0x00.
CRYPTO_IV_HEX	Input Parameter: User provided initialization vector (salt) submitted as a HEX value.	Should be 32 characters long AES and 16 characters long for 3DES. Short IVs are accepted and right filled with 0x00.
CRYPTO_DEBUG	Input Parameter: "yes" or blank	If set to "yes" configuration data will be displayed that may be helpful for debug purposes. Also for file encryption and decryption the contents (up to 100 bytes long) will be displayed.
CRYPTO_RC	Output Parameter: 0 – OK -1 – null or zero length input data! -2 – invalid key! -3 – invalid IV! -4 – invalid action! -5 – invalid option! -6 – encryption error! -7 – decryption error! -8 – license error! -9 – invalid BASE64 encoded input! -10 – invalid HEX encoded input! -11 – invalid action/option combination! -12 – invalid output file name! -13 – digest error! -14 – file not found! -15 – invalid HEX string!	A none zero return code indicates that an error has occurred.  Note: A '-5' return code generally means that you have not installed the "strong encryption" jar files or that they are not installed in the correct location. Please refer to section 3.2 of this document.
CRYPTO_ERROR_TEXT	Output Parameter: Text that briefly explains the error code	This value will be blank if an error did not occur.

## 2.2. ColdFusion Encryption Example:

The following ColdFusion code example results in the text "abcd" being encrypted using 256-bit AES encryption using PKCS5Padding in CBC mode. The encrypted value is returned in HEX format in the ColdFusion parameter CRYPTO\_OUT:

```
<cfset CRYPTO_ACTION = "5">
<cfset CRYPTO_OPTION = "104">
<cfset CRYPTO_IN = "abcd">
<cfset CRYPTO_KEY = "31313131313131313131313131313131">
<cfset CRYPTO_IV = "31313131313131313131313131313131">

<CFX_CRYPTO
```

---

```
CRYPTO_ACTION = #CRYPTO_ACTION#>
CRYPTO_OPTION = # CRYPTO_OPTION #>
CRYPTO_IN = #CRYPTO_IN#
CRYPTO_KEY = #CRYPTO_KEY#
CRYPTO_IV = #CRYPTO_IV#
>
```

```
<cfoutput>CRYPTO_OUT = #CRYPTO_OUT#<BR></cfoutput>
```

### 2.3. ColdFusion Decryption Example:

The following ColdFusion code example decrypts the value returned in the above example and returns the decrypted text in CRYPTO\_OUT:

```
<cfset CRYPTO_ACTION = "6">
<cfset CRYPTO_OPTION = "104">
<cfset CRYPTO_IN = #CRYPTO_OUT#>
<cfset CRYPTO_KEY = "31313131313131313131313131313131">
<cfset CRYPTO_IV = "31313131313131313131313131313131">
```

```
<CFX_CRYPTO
    CRYPTO_ACTION = #CRYPTO_ACTION#>
    CRYPTO_OPTION = # CRYPTO_OPTION #>
    CRYPTO_IN = #CRYPTO_IN#
    CRYPTO_KEY = #CRYPTO_KEY#
    CRYPTO_IV = #CRYPTO_IV#
>
<cfoutput>CRYPTO_OUT = #CRYPTO_OUT#<BR></cfoutput>
```

---

## 3. CRYPTOXPRESS CF INSTALLATION

### 3.1. Distribution Library

The following jar files are included in the distribution file (“CryptoXpressCF.zip”) for **CryptoXpress CF**:

File:	Description:
CFX_XPRESS..jar	Required jar file
local_policy.jar	Required jar file
US_export_policy.jar	Required jar file

### 3.2. Sample ColdFusion Programs

The following sample ColdFusion programs are included in the distribution zip file (“CryptoXpressCF.zip”) for **CryptoXpress CF**:

File:	Description:
Java_Xpress_Encrypt_File.cfm	Sample that encrypts and decrypts a file.
Java_Xpress_Encrypt_FileHex.cfm	Sample that encrypts and decrypts a file using a key and iv submitted in HEX format..
Java_Xpress_Encrypt_File_COMP.cfm	Sample that encrypts and decrypts a file using modes that create results that are compatible with CFXWorks’ original CFX_ENCRYPT_AES tag..
Java_Xpress_Encrypt_String.cfm	Sample that encrypts and decrypts a string.
Java_Xpress_Encrypt_String_COMP.cfm	Sample that encrypts and decrypts a string using modes that create results that are compatible with CFXWorks’ original CFX_ENCRYPT_AES tag..
Java_Xpress_MD5_File.cfm	Sample that creates a MD5 digest for a file.
Java_Xpress_MD5_String.cfm	Sample that creates a MD5 digest for a string.
Java_Xpress_SHA1_File.cfm	Sample that creates a SHA1 digest for a file.

Java_Xpress_SHA1_String.cfm	Sample that creates a SHA1 digest for a string.
Java_Xpress_Ascii2Hex.cfm	Sample that encodes an ASCII string in HEX then decodes it back to ASCII.
Java_Xpress_Ascii2Base64.cfm	Sample that encodes an ASCII string in BASE64 then decodes it back to ASCII.

### 3.3. Documentation

The following documentation is included in the distribution zip file (“CryptoXpressCF.zip”) for **CryptoXpress CF**:

File:	Description:
CryptoXpressCF.pdf	This document.
License_CryptoXpressCF.pdf	The license file.

### 3.4. CFX\_ENCRYPT\_AES Tag Compatibility

**The features described in this section should be ignored by all users who do not require compatibility with our original CFX\_ENCRYPT\_AES tag.**

The original CFXWorks encryption tag, CFX\_ENCRYPT\_AES, was developed and released in year 2000. At that time the implementation standards for AES were not completely defined. Today these standards exist and they are different then the guidelines as they existed in year 2000. The differences that exist are not in the AES algorithm itself but in how to handle padding, short keys, short IVs, and files. Therefore, we provide a compatibility feature in CFX\_XPRESS that encrypts and decrypts data precisely as we did in the original tag. We also provide some ready-to-run programs that can be used to encrypt and decrypt files encrypted by the original tag.

The following table maps the features we support from our original CFX\_ENCRYPT\_AES tag and indicates what features to code in our CFX\_XPRESS tag to achieve compatible results:

	CFX_ENCRYPT_AES PARAMETERS:		CFX_XPRESS PARAMETERS:	
128-bit	CBC=NO	COMP=NO	OPTION=901	COMP=NO
128-bit	CBC=NO	COMP=YES	OPTION =901	COMP=YES
128-bit	CBC=YES	COMP=NO	OPTION =903	COMP=NO
128-bit	CBC=YES	COMP=YES	OPTION =903	COMP=YES
256-bit	CBC=NO	COMP=NO	OPTION =902	COMP=NO
256-bit	CBC=NO	COMP=YES	OPTION =902	COMP=YES
256-bit	CBC=YES	COMP=NO	OPTION =904	COMP=NO
256-bit	CBC=YES	COMP=YES	OPTION =904	COMP=YES

The following Java ready-to-run Java programs are also provided that can be used to encrypt and decrypt files compatible with CFX\_ENCRYPT\_AES:

Java Programs	Description
CFX901FF	This Java program uses 128-bit AES encryption to encrypt or decrypt a file. It produces results compatible with OPTION=901 in the above table.
tag901.bat or xtag901 – sample program (1)	This script creates a file then uses CFX901FF to encrypt and decrypt it.
CFX902FF	This Java program uses 128-bit AES encryption to encrypt or decrypt a file. It produces results compatible with OPTION=902 in the above table.
tag902.bat or xtag902 – sample program (1)	This script creates a file then uses CFX902FF to encrypt and decrypt it.
CFX903FF	This Java program uses 128-bit AES encryption to encrypt or decrypt a file. It produces results compatible with OPTION=903 in the above table.
tag903.bat or xtag903 – sample program (1)	This script creates a file then uses CFX903FF to encrypt and decrypt it.
CFX904FF	This Java program uses 128-bit AES encryption to encrypt or decrypt a file. It produces results compatible with OPTION=904 in the above table.
tag904.bat or xtag904 – sample program (1)	This script creates a file then uses CFX904FF to encrypt and decrypt it.

(1) The .bat version is for Windows. The x version is for Linux, UNIX or the OS/400.

The input parameters supported for the above sample Java programs are illustrated in the following table.

Parameter	Description	Sample
/a	Encrypt or decrypt data (1)	/ae encrypt /ad – decrypt

/i	The fully qualified name of the input file (1)	/iinput.txt
/o	The fully qualified name of the output file (1)	/ioutput.txt
/k	The key to use for encryption/decryption. (1)	/k123
/v	The initialization vector to use for encryption/decryption. (1)	/v456
/c	Use COMP=YES or COMP=NO	/cy or /cn

(1) No defaults are assumed for any of these values.

### 3.5. .NET Compatibility

The .NET encryption routines support a very limited number of crypto options. Therefore, in some circumstances it will be impossible to use the .NET crypto routines provided by Microsoft to produce results that can be exchanged with non-.NET platforms. If however, your .NET code produces results consistent with the NIST test vectors and is implemented consistent with the following rules, the crypto results should be interchangeable with CryptoXpress CF developed and supported solutions. The rules include:

- Use either 128-bit or 256-bit AES
- Use a 16 byte block size
- Use CBC mode of operation
- Use PKCS5Padding
- For 128-bit encryption round short keys up to 16 bytes right filling them with 0x00.
- For 256-bit encryption round short keys up to 32 bytes right filling them with 0x00.
- Round short IVs up to 16 bytes right filling them with 0x00.

### 3.6. Installation on ColdFusion

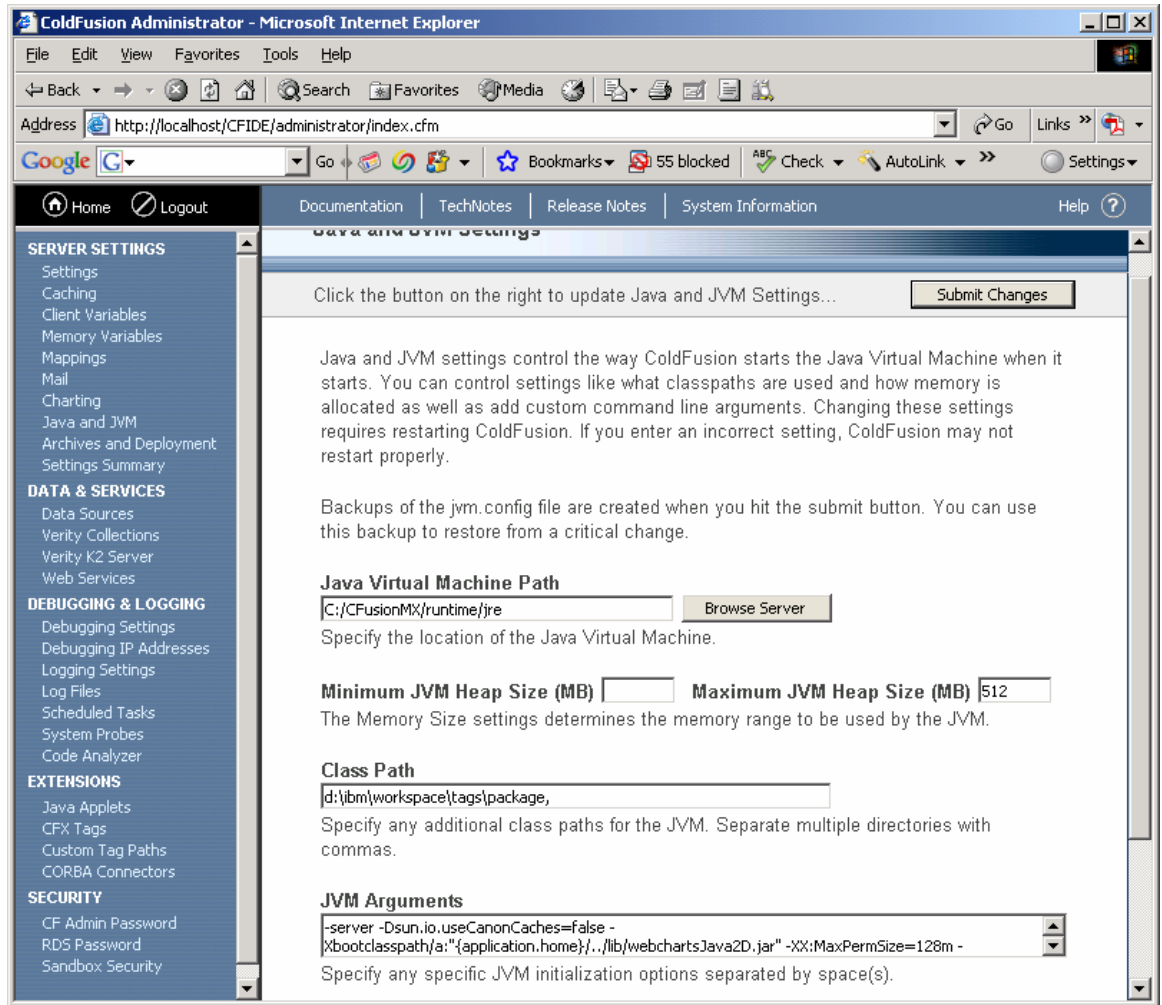
The **CryptoXpress CF** installation process is outlined below. Note that Java is case sensitive when defining class and jar file names. The most common cause of installation failure is failure to respect this Java discipline.

1. **CryptoXpress CF** is distributed in zip file (“CryptoXpressCF.zip”) that contains Java jar files, this document, Java example programs and sample ColdFusion programs. The command you must enter at a command line to unzip this file is as follows:

---

pkunzip filename

You can also use WinZip.



*Figure 5 - Registering CFX\_XPRESS*

The actual text added to the Classpath doesn't display well in the above figure. Assuming the directory that contains the jar files is "d:\IBM\workspace\Tags\package, the actual text added to the class path would be:

Windows

"d:\IBM\workspace\Tags\package"

**You must stop and restart the ColdFusion server for these changes to take place.**

2. ColdFusion also requires that you register the tag using the ColdFusion Administrator. The registration dialog will look similar to the following:

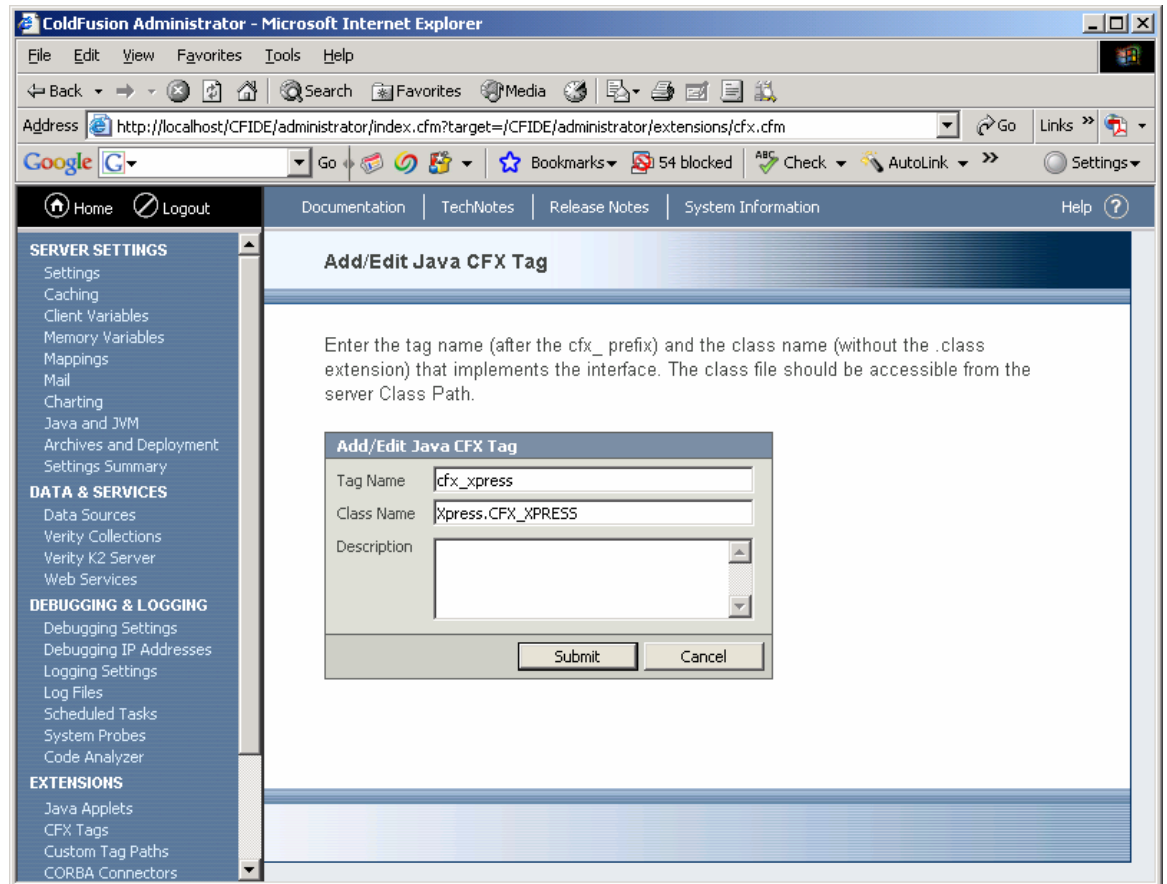


Figure 6 - Registering CryptoXpress CF

Name the tag whatever you like. The sample programs use the ColdFusion tag name "CFX\_XPRESS". Next, you must tell ColdFusion the name of the Java class to execute when the tag is called. **Java is very unforgiving if you make an error entering this value. The value is case sensitive.** You must enter the following "Xpress.CFX\_XPRESS".

3. If you want to use the encryption capabilities provided by this tag, you must copy the "strong encryption" jar files available from Sun to the appropriate directory on the host system. Without these jar files the encryption capabilities provided by this tag will not work. There are two options as to where these files must be installed.

---

### Option 1

If ColdFusion is installed without “JAVA\_HOME” set to point to the Version of Java you are executing at run time, then you must install the files, (local\_policy.jar and US\_export\_policy.jar) in the “/ColdFusion/runtime/jre/lib/security” directory. They replace files of the same name that are distributed by ColdFusion. Please substitute “ColdFusion” with whatever your ColdFusion installation directory is. The ColdFusion copy of these files allow for 128-bit encryption but not 256-bit. A copy of the “strong encryption” jar files is included in the “CryptoXpressCF.zip” file.

### Option 2

If ColdFusion is installed with “JAVA\_HOME” set to point to a runtime or SDK Version of Java. These files, (local\_policy.jar and US\_export\_policy.jar) must be installed in the %JAVA\_HOME%/jre/lib/security directory. They replace files of the same name that are distributed in the standard Java SDK. The standard distribution files do not allow the use of strong encryption. A copy of these jar files is include in the distribution zip file.

On a Windows system “JAVA\_HOME” can be found by entering the command:

```
Echo %JAVA_HOME%
```

On a Solaris or Linux system “JAVA\_HOME” can be found by entering the command:

```
Echo $JAVA_HOME$
```

That’s it! The entire installation process should take only a few minutes to complete.

## **3.7. Minimum System Requirements**

This tag has been tested on Windows, Linux, and Solaris platforms. It should function properly on any system executing ColdFusion Release 6.0 of later. It has been tested successfully on ColdFusion 6.x ,7.x and 8.0.

## **3.8. Export Limitations**

---

This tag software contains encryption technology that is subject to the U.S. Export Administration Regulations and other U.S. law, and may not be exported or re-exported to certain countries (currently Afghanistan (Taliban-controlled areas), Cuba, Iran, Iraq, Libya, North Korea, Serbia (except Kosovo), Sudan and Syria) or to persons or entities prohibited from receiving U.S. exports (including Denied Parties, entities on the Bureau of Export Administration Entity List, and Specially Designated Nationals). For more information on the U.S. Export Administration Regulations <http://www.bxa.doc.gov/Encryption/regs.htm>, 15 C.F.R. Parts 730-774, and the Bureau of Export Administration U. S. Department of Commerce. Please see the home page [www.bxa.doc.gov](http://www.bxa.doc.gov)