

---

# CFX\_BASE64™

A Base64 Solution for ColdFusion® Users



## Installation & User Guide

For Windows, Linux & Solaris

Software Version	3.0
Document	cfxbase64.pdf
Published	02/12/2003

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

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

<http://www.CFXWorks-Coldfusion.com>

Printed in the United States of America.

---

<b>1. INTRODUCTION</b> .....	<b>3</b>
1.1.    THANK YOU.....	3
1.2.    OVERVIEW.....	3
1.3.    THE BASE64 ENCODING STANDARD.....	3
1.4.    QUALIFICATION.....	5
<b>2. DISCUSSION</b> .....	<b>5</b>
2.1.    WHY USE BASE64 ENCODING?.....	5
2.2.    WHY USE CFX_BASE64 VERSUS TOBASE64?.....	5
2.3.    HOW SECURE IS BASE64 ENCODING?.....	6
2.4.    WHAT IF I FORGET MY ENCRYPTION KEY?.....	6
<b>3. TECHNICAL ISSUES</b> .....	<b>7</b>
3.1.    TAG SPECIFICATION.....	7
3.2.    INSTALLATION.....	8
3.3.    PERFORMANCE ISSUES.....	9
3.4.    MESSAGE DIGESTS.....	10
3.5.    MINIMUM SYSTEM REQUIREMENTS.....	10
<b>4. OTHER STUFF</b> .....	<b>11</b>
4.1.    SOFTWARE LICENSE.....	11
4.2.    TECHNICAL LIMITATIONS.....	11
4.3.    EXPORT LIMITATIONS.....	11
4.4.    FUTURE POSSIBLE EXTENSIONS.....	11
4.5.    SUPPORT.....	11
4.6.    COPYRIGHT.....	11
4.7.    WARRANTY.....	12
<b>APPENDIX A. TAG PARAMETERS</b> .....	<b>13</b>
<b>APPENDIX B. ERROR CODES</b> .....	<b>15</b>
<b>APPENDIX C. LOG</b> .....	<b>17</b>
<b>APPENDIX D. SAMPLES</b> .....	<b>18</b>

---

# 1. INTRODUCTION

## 1.1. Thank You

Thank you for purchasing this product. As the author of CFX\_BASE64, it is my intent to develop an offering that adds value to your ColdFusion efforts by improving your ability to develop secure solutions. If you feel a need to contact me directly, please send an email to [anickles@cfxworks.com](mailto:anickles@cfxworks.com).

## 1.2. Overview

CFX\_BASE64 is a ColdFusion CFX tag that encodes and decodes text and files (ASCII and binary) using the Base64 algorithm as defined in RFC 1521. Please see <ftp://ftp.isi.edu/in-notes/rfc1521.txt>

## 1.3. The Base64 Encoding Standard

The following description is extracted directly from RFC 1521:

“The Base64 Content-Transfer-Encoding is designed to represent arbitrary sequences of octets in a form that need not be humanly readable. The encoding and decoding algorithms are simple, but the encoded data are consistently only about 33 percent larger than the unencoded data. This encoding is virtually identical to the one used in Privacy Enhanced Mail applications, as defined in RFC 1421. The Base64 encoding is adapted from RFC 1421, with one change: Base64 eliminates the "\*" mechanism for embedded clear text.”

“A 65-character subset of US-ASCII is used, enabling 6 bits to be represented per printable character. (The extra 65th character, "=", is used to signify a special processing function.)”

“NOTE: This subset has the important property that it is represented identically in all versions of ISO 646, including US ASCII, and all characters in the subset are also represented identically in all versions of EBCDIC. Other popular encodings, such as the encoding used by the UUENCODE utility and the base85 encoding specified as part of Level 2 PostScript, do not share these properties, and thus do not fulfill the portability requirements a binary transport encoding for mail must meet.”

“The encoding process represents 24-bit groups of input bits as output strings of 4 encoded characters. Proceeding from left to right, a 24-bit

input group is formed by concatenating 3 8-bit input groups. These 24 bits are then treated as 4 concatenated 6-bit groups, each of which is translated into a single digit in the Base64 alphabet. When encoding a bit stream via the Base64 encoding, the bit stream must be presumed to be ordered with the most- significant-bit first.”

“That is, the first bit in the stream will be the high-order bit in the first byte, and the eighth bit will be the low-order bit in the first byte, and so on.”

“Each 6-bit group is used as an index into an array of 64 printable characters. The character referenced by the index is placed in the output string. These characters, identified in Table 1, below, are selected so as to be universally representable, and the set excludes characters with particular significance to SMTP (e.g., ".", "CR", "LF") and to the encapsulation boundaries defined in this document (e.g., "-").”

Table 1: The Base64 Alphabet

<u>Value</u>	<u>Encoding</u>	<u>Value</u>	<u>Encoding</u>	<u>Value</u>	<u>Encoding</u>	<u>Value</u>	<u>Encoding</u>
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

“Special processing is performed if fewer than 24 bits are available at the end of the data being encoded. A full encoding quantum is always completed at the end of a body. When fewer than 24 input bits are available in an input group, zero bits are added (on the right) to form an integral number of 6-bit groups. Output character positions which are not required to represent actual input data are set to the character “=”. Since all Base64 input is an integral number of octets, only the following cases can arise: (1) the final quantum of encoding input is an integral multiple of 24

---

bits; here, the final unit of encoded output will be an integral multiple of 4 characters with no "=" padding, (2) the final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output will be two characters followed by two "=" padding characters, or (3) the final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output will be three characters followed by one "=" padding character."

## 1.4. Qualification

This release of the CFX\_BASE64 tag does "not" implement the following provision of the Base64 specification:

"The output stream (encoded bytes) must be represented in lines of no more than 76 characters each. All line breaks or other characters not found in Table 1 must be ignored by decoding software. In Base64 data, characters other than those in Table 1, line breaks, and other white space probably indicate a transmission error, about which a warning message or even a message rejection might be appropriate under some circumstances."

As I reviewed other implementations of the Base64 encoding algorithm, I found that this provision of the specification was generally ignored.

## 2. DISCUSSION

### 2.1. Why Use Base64 Encoding?

The most common use of Base64 encoding is to encode binary and text data that may contain non-ASCII characters, or characters that are considered special or reserved characters so that this information can be safely transmitted across the Internet. All binary representations, including 0x00 - 0xFF, can be Base64 encoded. It is particularly useful to users of XML and SOAP because the information one wishes to enclose within these documents may contain reserved or special characters.

Base64 encoded data can also be printed. This is not true for many valid data characters.

Some use is also made of Base64 encoding to hide clear text from prying eyes. Base64, however, should not be used as a substitute for encryption because this encoding technique does not use a key.

### 2.2. Why use CFX\_BASE64 versus toBase64?

---

Unfortunately, there are several errors in ColdFusion's implementation of the "toBase64" function. Although some of these errors exist in the Windows implementation, they are especially prevalent in the Linux and Solaris implementation. Also, ColdFusion's "toBase64" **function is not implemented consistently across the Windows, Linux and Solaris platforms**. Also, CFX\_BASE64 supports many capabilities not supported by the toBase64 function including:

1. encoding files
2. decoding files
3. encoding text to a file
4. decoding text to a file
5. support for message digests

## 2.3. How Secure Is Base64 Encoding?

Base64 encoding should not be considered secure. It should not be used as a substitute for encryption.

## 2.4. What if I forget my encryption key?

Base64 does not use an encryption key. There is no key to forget.

---

## 3. TECHNICAL ISSUES

### 3.1. Tag Specification

The programming specification for the CFX\_BASE64 tag is as follows:

<CFX_BASE64		Default
ACTION=		
"et"	encode text	et
"dt"	decode text	
"ef"	encode file	
"df"	decode file	
"etf"	encode text to file	
"dft"	decode text from file	
DIGEST=		
"yes or no"	result contains a MD5 message digest.	no
TEXT=		
"text"	input if action = et,dt,etf or df	none
NAME=		
"RESULT"	output if action = et,dt,dft	"TEXT"
FILEIN=		
"filein"	input file if action = ef,df, or dft	"filein.txt"
FILEOUT=		
"fileout"	output file if action = ef,d or etf	"fileout.txt"
DISPLAY=		
0,1 or 2	0 - Display nothing 1 - Display Log 2 - Write log to disk file "aes.log"	0
>		
Reply:		
RESPONSE	see Appendix B	
RC=	see Appendix B 0 -Zero length input n - Length of output data or file - 1_Invalid ACTION - 2 Not used by this tag - 3 FILEIN error - 4 FILEOUT error - 5 Memory error - 6 Data integrity error - 7 NAME error - 99 License error	

*Figure 1 - ColdFusion Tag*

---

For a detailed discussion of CFX\_BASE64 parameters, please see *Appendix A. Tag Parameters*. For a detailed discussion of CFX\_BASE64 error codes please see *Appendix B. Error Codes*. For a discussion of the contents of the log if “DISPLAY=2” is specified, please see *Appendix C. Log*. For specific examples of how to deploy this tag, please see *Appendix D. Samples*.

## 3.2. Installation

The CFX\_BASE64 installation process is a simple three-step process:

1. CFX\_BASE64 is distributed as zip file (filename.zip). The actual filename is specific to each licensee. You will need a “password” to unzip this file. The password is sent to the licensee by email when the tag is ordered. The command you must enter at a command line is as follows:

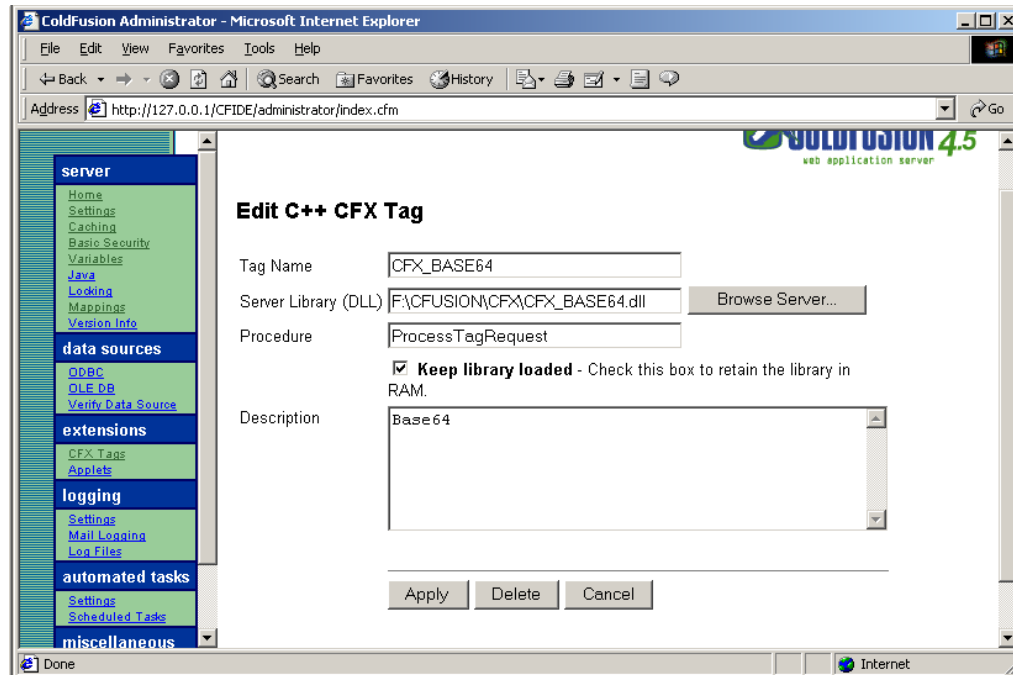
Windows:	pkunzip filename -s[password]
Linux:	unzip -j filename
Solaris	unzip -j filename

Note that commands and filenames in Linux and Solaris are case sensitive. The “-j” in the above Linux and Solaris commands stand for a”junk“, or do not make directories. Without this parameter, the unzip command will create subdirectories within the current working directory. On Linux and Solaris, after you have unzipped the zip file, you will have to use the “chmod u+x cfx\_checkemail.so” command to make this file executable.

When you unzip this file, the code and documentation is extracted to the current working directory. The zip file contains the following files:

- CFX\_BASE64.dll (Windows) or “cfx\_base64.so” (Linux and Solaris) – the executable.
  - BASE640.cfm – sample program text-2-text encoding.
  - BASE641.cfm – sample program file-2-file encoding.
  - BASE642.cfm – sample program text-2-file encoding.
  - LICENSE\_BASE64.pdf – a copy of the software license.
  - CFXBASE64.pdf – this document.
2. On versions of Coldfusion prior to MX, ColdFusion recommends that you copy the CFX\_BASE64 executable to the \CFUSION\CFX directory for execution. This wasn’t absolutely necessary, but it was best practices. On Windows versions of MX, this directory would be \CFusionMX\CustomTags\cfx. On Linux and Solaris, this directory would be /opt/coldfusionmx/cfx.

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



*Figure 2- Registering CFX\_BASE64*

You should name the tag as shown in the *Figure 2*. You must tell ColdFusion where the executable is stored. For performance reasons you should select “Keep library loaded” however this is not required if the tag will not be in heavy use. You can enter a description as shown above. For Windows, if you do not check the “Keep library loaded” checkbox, the dll is reloaded each time the tag is executed. Therefore, if you install a new version of the dll, it is automatically refreshed at runtime. On Linux and Solaris, if you install a new version of the executable, you must stop and start the Coldfusion server to refresh the executable.

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

### 3.3. Performance Issues

One of the major issues with all encryption and encoding algorithms relates to performance. CFXWorks has performed extensive performance measurements on many encryption algorithms and encoding techniques with varying message

---

lengths, key sizes, and block sizes. We also tested across several development platforms and environments. We believe that in a ColdFusion environment, the following issues should be considered relative to performance:

- The primary issue impacting performance will be the selection of the installation option “Keep library loaded”, see *Figure 2*. If this option is not selected, the CFX\_BASE64 dll is loaded each time the tag is referenced. Therefore, the major issue impacting performance of the tag will be I/O speed.
- The next issue impacting performance will be the length of data being encoded. The longer the data string or file, the more CPU cycles will be consumed performing encoding and decoding.

### 3.4. Message Digests

Encryption is intended to protect the confidentiality of data. However, how do you determine if a black hat (bad guy) has changed the contents of an encrypted data string or data file? Changing data content relates to data integrity, not data confidentiality. The solution to this issue is what cryptologist calls message digests. CFX\_BASE64 uses MD5, a best of breed message digest algorithm, to calculate message digests. If “DIGEST=YES” is coded, a message digest is calculated and concatenated to the data string. Then, the total string is encrypted. This forms a digital envelope. When the data is decrypted, the message digest is recalculated and compared to the original value. If even a single bit within the encrypted string or file has been modified, the comparison fails and CFX\_BASE64 returns an error code (-6 data integrity error). Please see *Appendix B. Error Codes*.

### 3.5. Minimum System Requirements

These offerings have been tested on Windows 2000, Linux (Red Hat) and Solaris 8 platforms. The Windows version should function properly on all Win32 platforms. The Linux version should function on all Intel versions of Linux. The Solaris offering should function on Solaris 8 and 9.

The ColdFusion tag versions of this offering requires that it be installed on a system using a properly configured copy of ColdFusion 4.0, 4.5, 5.0 or MX.

---

## 4. OTHER STUFF

### 4.1. Software License

Demonstration tags contain expiration dates that expire in 30 days. Purchased tags have no expiration date. The tag is licensed on a per system basis. The license agreement is contained in the product distribution zip file. Please read this pdf file for detailed license information.

Encoding will **not** be performed and a message digest will **not** be created on systems whose license has expired. In other words, if the license has expired, the output of the tag will be an image of the input. If the input is in clear text, the output will be in clear text.

### 4.2. Technical Limitations

CFX\_BASE64 has been tested extensively on message strings from 0-65536 characters in length and for file sizes up to 2 Mbytes in size. There is no technical reason we know of, that messages or files exceeding these values will not function properly, however, we do not test beyond these limitations.

### 4.3. Export Limitations

This tag is not subject to export restrictions.

### 4.4. Future Possible Extensions

Please send us your suggestions. We are particularly interested in your feedback as to creative ways to use the CFX\_BASE64 capability.

### 4.5. Support

Support is provided for this offering via the CFXWorks web site [www.CFXWorks-Coldfusion.com](http://www.CFXWorks-Coldfusion.com). A Q&A forum is maintained and CFXWorks will attempt to respond to emails addressed to [support@cfxworks.com](mailto:support@cfxworks.com) on this forum.

### 4.6. Copyright

---

The CFX\_BASE64 code and documentation is Copyrighted.

## **4.7. Warranty**

Please read the license file “license\_base64.pdf” in the distribution zip file.

---

## APPENDIX A. TAG PARAMETERS

<u>Parameter</u>	<u>Default Value</u>	<u>Required</u>	<u>Description</u>
ACTION =	“et”	optional	Encode text.
	“dt”		Decode text.
	“ef”		Encode file.
	“df”		Decode file.
	“etf”		Encode text to file.
	“dft”		Decode file to text.
DIGEST = “yes” or “no”	“no”	optional	Yes – add message digest. No – Don’t add digest.
TEXT = “Text to be encrypted”	none	required for “et” & “etf”	Null terminated text string.
NAME = “RESULT”	“TEXT”	optional	Null terminated text string.
FILEIN = “filename”	“filein.txt”	required for “ef”, “df” & “dft”	Fully qualified file name (2).
FILEOUT = “filename”	“fileout.txt”	required for “ef”, “df” & “etf”	Fully qualified file name(2).
DISPLAY = value	0	optional	0 – no display 1 – display log 2 – create a disk log (2).

(1) If a fully qualified filename is not specified, the file is read from and written to the current working directory for ColdFusion. This directory most likely is c:\winnt\system32. If you execute the tag using DISPLAY=1, the current working directory will be displayed.

(2) The log is written to the current working directory of ColdFusion using the filename “base64.log”.

*Figure 3* provides a cross reference between the “ACTION” selected and the parameters supported.

ACTION PARAMETER	ET	DT	EF	DF	ETF	DFT
ACTION	D	O	O	O	O	O
DIGEST	O	O	O	O	O	O
TEXT	R	R	NA	NA	R	X
NAME	O	O	NA	NA	NA	O
FILEIN	NA	NA	R	R	NA	R
FILEOUT	NA	NA	R	R	R	NA
DISPLAY	O	O	O	O	O	O
RESPONSE	X	X	X	X	X	X
RC	X	X	X	X	X	X

D - Default Value  
 O - Optional Parameter  
 R - Required Parameter  
 X - Output Value  
 NA - Not Applicable

*Figure 3 - Tag Parameters Versus ACTION*

---

## APPENDIX B. ERROR CODES

<u>RC</u>	<u>Description</u>	<u>Action of tag</u>
0	A return code of 0 indicates that the length of the input string or data file is 0.	Output is created with a zero length. For example with a text value of "" or with a file of zero length.
n	A positive value indicates that the encoding or decoding task has been successful.	The value is the length of the output text or file.
-1	Invalid ACTION parameter.	The tag takes no action.
-2	Not used by this tag.	
-3	FILIIN error. Most likely the file cannot be found or the directory specified does not exist.	The status of the output text or file is unknown.
-4	FILEOUT error. Most likely the directory or drive specified does not exist.	The status of the output text or file is unknown.
-5	Memory allocation error. This means that you do not have enough memory on your system to complete the encoding/decoding task.	The input is copied to the output unchanged.
-6	Data integrity error. This means that the contents of an encoded data string or file has been modified.	No output is created.
-7	Invalid NAME. This means that the variable name passed to the tag contained blanks or non-ASCII characters.	No output is created.
-99	License error. This means that an attempt to execute the tag has occurred on an unlicensed system. The tag is executed, however, no encoding or decoding is performed.	The input is copied to the output unchanged

*Figure 4* provides a cross reference between return codes and actions.

ACTION RETURN CODE		VALUE	ACTION					DFT
			ET	DT	EF	DF	ETF	
Zero length input	(1)	0	x	x	x	x	x	x
Length of output	(2)	n	x	x	x	x	x	x
Invalid ACTION		-1	x	x	x	x	x	x
Invalid KEY		-2	x	x	x	x	x	x
FILEIN error	(3)	-3			x	x		x
FILEOUT error		-4			x	x	x	
Memory error		-5	x	x	x	x	x	x
Data integrity error		-6		x		x		x
NAME error		-7	x	x				x
Clear text passthru	(4)	-99	x	x	x	x	x	x
x - Possible RC			"blank" - not possible for this ACTION					

*Figure 4 – Error Codes Versus ACTION*

---

## APPENDIX C. LOG

The “DISPLAY=2” option causes CFX\_BASE64 to create a log. This option should be selected only for debugging purposes. It will **seriously** degrade performance. *Figure 5* reflects what is written to the log for each of the ACTIONS selected.

LOG CONTENT DISPLAY=2	ET	DT	EF	DF	ETF	DFT
Setup data	x	x	x	x	x	x
License data	x	x	x	x	x	x
Tag parameters	x	x	x	x	x	x
Input data	x	x			x	
Output data	x	x				x

Users who have problems should eMail a copy of their log, and a copy of both the input and output files, if applicable, to [service@CFXWorks.com](mailto:service@CFXWorks.com).

*Figure 5 - Log Content Versus ACTION*

---

## APPENDIX D. SAMPLES

*Figure 6* illustrates just how simple it is to use this tag.

```
Example:
<CFX_BASE64
  ACTION="et"
  TEXT="This is the data to be encrypted"
  NAME="RESULT">

Example:
<CFX_BASE64
  ACTION="ef"
  FILEIN="afile"
  FILEOUT="bfile">

Example:
<CFX_BASE64
  ACTION="etf"
  TEXT="This is the data to be encrypted"
  FILEOUT="bfile">
```

*Figure 6 - Samples*

The CFX\_BASE64 distribution files contain the following sample programs.

- “base640.cfm”      This sample program encodes and decodes text strings.
- “base641.cfm”      This sample program encodes file-to-file and performs the reverse operation. This example assumes the presence of some directories and a file that may not be valid on your system. Please modify these values to the correct values for your system.
- “base642.cfm”      This sample program encodes a text string to a file and performs the reverse operation. This example assumes the presence of a directory that may not be valid on your system. Please modify these values to the correct values for your system.