Table of Contents

Introduction

Welcome to the Alagad Captcha documentation. This documentation is intended to help you integrate the Alagad Captcha Component into your ColdFusion application.

The Alagad Captcha is a ColdFusion Component (CFC) written in 100% native ColdFusion which generates images of obfuscated text. The text is intended to be human readable but not machine readable. By comparing a user provided string to the known value of the string displayed in the image, you can confirm that a human, not a computer, is interacting with your application.

Captcha is an acronym for "**c**ompletely **a**utomated **p**ublic **T**uring test to tell **c**omputers and **h**umans **a**part". Captcha images tend to be used on web forms to prevent automated submission by computers. More information on Captchas can be found at http://en.wikipedia.org/wiki/Captcha.

This component requires no third party software to operate and no additional configuration.

This documentation assumes that you have some experience with ColdFusion and CFCs.

Conventions Used In This Document:

Words surrounded in angle brackets indicate ColdFusion or HTML tags. For example, <cfset> indicates the ColdFusion cfset tag.

Words highlighted in blue followed by parenthesis indicate Alagad Captcha methods. For example createCaptcha() indicates the createCaptcha method.

ColdFusion and HTML code is highlighted in brown. For example:

```
<cfset example="Hello World" />
```

Paths to files or directories are surrounded in quotation marks. For example "/MyDirectory/".

The "¬" character at the end of a line of example code indicates a line which should be continued on the same line, even though it's shown as wrapping.

About the Alagad Captcha

The Alagad Captcha is a ColdFusion Component (CFC) written in 100% native ColdFusion which generates images of obfuscated text. The text is intended to be human readable but not machine readable. By comparing a user provided string to the known value of the string displayed in the image, you can confirm that a human, not a computer, is interacting with your application.

Captcha is an acronym for "**c**ompletely **a**utomated **p**ublic **T**uring test to tell **c**omputers and **h**umans **a**part". Captcha images tend to be used on web forms to prevent automated submission by computers. More information on Captchas can be found at http://en.wikipedia.org/wiki/Captcha.

The Alagad Captcha component requires no third party software to operate and no additional configuration.

The following is an example of an image generated by Alagad Captcha:

Because the Alagad Captcha is written in pure ColdFusion and instantiates only native Java objects, it compiles with the rest of your CFML files to Java bytecode.

The Alagad Captcha requires almost no effort to install and use. Simply place the component into your custom tags directory or any directory in your site and then instantiate it using the ColdFusion CreateObject method. Once you have the Captcha component instantiated you can use its methods to begin generating Captcha images.

Requirements to Use the Alagad Captcha

The Alagad Captcha works with ColdFusion MX and later on any supported platform.

ColdFusion MX 6.1 users will not need to do any additional configuration for the Alagad Captcha to work.

Users who are running ColdFusion MX without the 6.1 update will need to insure that they are running version 1.4 or later the Java Runtime Environment (JRE).

Click here for information on updating your JRE.

The Captcha component no longer supports Blue Dragon.

For the Alagad Captcha component to function, the ColdFusion function CreateObject() must be enabled.

Updating your JRE

By default, ColdFusion MX is installed with version 1.3.1 of the Java Runtime Environment (JRE). Users who are running ColdFusion MX without the 6.1 update will need to insure that they are running a version 1.4.1 or later the JRE to use the Alagad Captcha. Alagad suggests using the most recent JRE.

To update your JRE:

1. Download and install the JRE from http://java.sun.com.
2. Using the ColdFusion Administrator, set your Java Virtual Machine Path to the path to the JRE. For example: C:\Program Files\Java\j2re1.4.2_04\. For more information see the ColdFusion documentation.
3. Restart ColdFusion.

You should now be able to use the Alagad Captcha on your ColdFusion MX server.

Strange Error Messages and Headless Systems

Error: This graphics environment can be used only in the software emulation mode

Many servers do not have a mouse, keyboard or a display. Java considers these systems to be "headless". The Alagad Captcha may not run correctly without additional configuration in these systems. In these circumstances, when you try to generate Captcha images you may receive this error: *This graphics environment can be used only in the software emulation mode*.

Macromedia has posted a TechNote 18747 which addresses these problems at http://www.macromedia.com/support/coldfusion/ts/documents/graphics_unix_141jvm.htm

Unfortunately, the solution to the problem causes the <CFchart> tag not to function.

To solve the problem:

1) Stop ColdFusion

2) Edit the file /cf_root/runtime/bin/jvm.config.

3) Change the line which starts with java.args=…. So that "-Djava.awt.graphicsenv=com.gp.java2d.ExHeadlessGraphicsEnvironment" is changed to "-Djava.awt.headless=true".

   Example:

```
java.args=-server -Xmx512m -Dsun.io.useCanonCaches=false
-Xbootclasspath/a:{application.home}/lib/webchartsJava2D.jar
-XX:MaxPermSize=128m -XX:+UseParallelGC
-Djava.awt.headless=true
```

4) Restart ColdFusion

Systems primarily affected by this problem are: Linux, HPUX, Solaris.

Additional information can be seen on the personal blog of Doug Hughes at http://www.doughughes.net/index.cfm/page-blogLink/entryId-29.

Installation of the Alagad Captcha

Installation of the Alagad Captcha is quite simple. First download the Alagad Captcha in .zip format from http://www.alagad.com. Once you have the Captcha component, extract the Captcha.cfc file from the Zip file from the correct folder. If you are using ColdFusion server you will want to extract the Captcha.cfc file from "ColdFusion MX and MX 6.1 Version" folder.

If you want to use the Captcha component for any website on your server you can please the Captcha.cfc file in any custom tags folder.

If you want to use the Captcha component in only one website, copy the Captcha.cfc file into the website's root or any directory under your web site's root.

If you want to place the Captcha component outside of your web site directory hierarchy and out side of any of custom tags folder, then simple copy the Captcha.cfc file anywhere you want on your file system and create a mapping to that directory using the ColdFusion administration interface. For more information see the ColdFusion documentation.

Alagad Captcha License Keys

The Alagad Captcha component is protected via a "nagging" system. If you are using the Captcha component without providing a license key or with an invalid license key you will see the text "UNLICENSED" in your Captcha instead of any provided or random text.

Use your license key to remove this text. For information on using your license key see Using Your License Key.

Instantiating the Alagad Captcha

You must instantiate the Alagad Captcha before you can call its methods. The following examples demonstrate instantiating the Alagad Captcha using the ColdFusion <cfobject> tag and CreateObject() function.

For more information on using ColdFusion components see the Using ColdFusion components section of the ColdFusion documentation.

Instantiation from a Local Directory

If you installed the Alagad Captcha in a directory under your web site's root you would instantiate it as follows below. In both cases we are assuming the Captcha.cfc file was placed in a directory "/path/to/" under your web site's root. We are creating an instance of the Captcha.cfc named myCaptcha

Using <cfobject>:

```
<cfobject component="path.to.Captcha" name="myCaptcha" />
```

Using CreateObject():

```
<cfset myCaptcha = CreateObject("Component", "path.to.Captcha") />
```

Instantiation from a Mapped Directory

If you installed the Alagad Captcha in a ColdFusion mapped directory you would instantiate it as follows below. In both cases we are assuming the Captcha.cfc file was placed in a directory which was mapped to with the logical path "/mapped/path/to/". We are creating an instance of the Captcha.cfc named myCaptcha.

For more information on ColdFusion mappings see the ColdFusion documentation.

Using <cfobject>:

```
<cfobject component="mapped.path.to.Captcha" name="myCaptcha" />
```

Using CreateObject():

```
<cfset myCaptcha = CreateObject("Component", "mapped.path.to.Captcha") />
```

Instantiation from a Custom Tag Path

If you installed the Alagad Captcha in a ColdFusion custom tags path you would instantiate it as follows below. In both cases we are creating an instance of the Captcha.cfc named myCaptcha.

Using <cfobject>:

```
<cfobject component="Captcha" name="myCaptcha" />
```

Using CreateObject():

```
<cfset myCaptcha = CreateObject("Component", "Captcha") />
```

Using Your License Key

License keys are 29 character strings in the following format:

XXXXX-XXXXX-XXXXX-XXXXX-XXXXX

To use the license you must pass the license key into the configure() method when it is called or create a file named captchakey.txt in the same directory as your Captcha.cfc file.

Example, using the configure() method

The following code demonstrates two techniques for invoking the Captcha component, calling the configure() method and passing in the license key.

Using <cfobject>:

```
<cfobject component="path.to.Captcha" name="myCaptcha" />
<cfset myCaptcha.configure(expandPath("."), ¬
"XXXXX-XXXXX-XXXXX-XXXXX-XXXXX") />
```

Using CreateObject():

```
<cfset myCaptcha = CreateObject("Component", ¬
"path.to.Captcha").configure(expandPath("."), ¬
"XXXXX-XXXXX-XXXXX-XXXXX-XXXXX") />
```

When you pass a valid license key to the configure() method the “UNLICENSED” text will be removed and a Captcha string will be displayed instead.

Using captchakey.txt File

By using the captchakey.txt file you can automatically set the license key instead of passing it to every call of the configure() method. This method is the way in which hosing providers should make the Captcha component available to their users.

To use this, create a file named captchakey.txt which contains only the text of your license key. Place the captchakey.txt file into the same directory as your Captcha.cfc file. When the component is invoked it will automatically find this file and import the key. Hosting providers should place the Captcha.cfc and captchakey.txt file in a folder under their custom tags directory.

If the key is not passed to the configure() method or invalid key is passed to the configure() method, or the captchakey.txt file is not provided or has an invalid key, the “UNLICENSED” text will be displayed.

All examples in this documentation are shown with the key parameter being provided to the configure() method.

Using the Instantiated Alagad Captcha Component

Once you have instantiated the Captcha component you can call methods on the component to generate Captcha images. A complete example is provided in the Example directory in the downloaded Zip file.

The examples use the ColdFusion version of the Captcha component.

Alagad Captcha Methods

The Alagad Captcha component has only 6 methods to use when creating Captcha images.

The following is a list of Alagad Captcha methods. Methods are listed in the order of importance.

configure()

Description

The configure() method is used to configure the Captcha component. This method must be called when invoking the component or directly after invoking the component.

The configure() method requires the directory attribute to be provided. This attribute instructs the Captcha component on where to output Captcha images.

If you have registered the Alagad Captcha and are not using the captchakey.txt file then you will need to pass your key into the key attribute when invoking the component. This will register the component and remove the "UNLICENSED" text.

The configure() method returns the configured component.

Syntax

Captcha = configure(directory, key)

Parameter	Required	Type	Description
Directory	Yes	String	The directory to output Captcha images to.
Key	No	String	Your Captcha license key, if available and not already provided in the captchakey.txt file.

Example

The following two examples instantiate the Captcha component, set the output directory to the current directory, and pass in the license key.

Example 1, all on one line:

```
<cfset myCaptcha = CreateObject("Component", ¬
"Captcha").configure(expandPath("."), "XXXXX-XXXXX-XXXXX-XXXXX-XXXXX")
/>
```

Example 2, on two lines:

```
<cfset myCaptcha = CreateObject("Component", "Captcha") />
<cfset myCaptcha.configure(expandPath("."), ¬
"XXXXX-XXXXX-XXXXX-XXXXX-XXXXX") />
```

createCaptcha()

Description

The createCaptcha() method is used to generate the Captcha image. The method returns a structure of information about the generated image's contents.

The structure contains the following elements:

Directory
This is the directory where images are written to. This is the same as the value provided when calling the configure() method.

FileName
This is the name of the generated file. This can either be the value of the file name attribute or a random file name. This is useful to determine which image was generated.

FontsUsed
This is an array of the fonts used to create the image. Some fonts contain characters which can not be displayed. These characters are displayed as rectangles. Because of this, your users will not know what character was displayed. In this case you can use the FontsUsed element to determine the font of a specific character and then use the setIgnoredFontList() method to prevent this font from being used. Elements in this array correspond to the characters displayed in the Captcha. (Element 1 is the font used for character 1.)

Hash
This is a hash of a lowercase version of the string displayed. This should be passed to the validate() method to confirm that the text provided by the user matches the Captcha image contents.

String
This is the plain text of the string displayed in the Captcha image.

If your have not provided a valid license key the resulting Captcha image will always contain the text "UNLICENSED", even when you provide the string attribute for this method. Once the key is provided this method will either generate a random string for you or use the one provided to the string attribute.

Syntax

captchaStructure = createCaptcha(filename, string)

Parameter	Required	Type	Description
FileName	No	String	This is the name of the file to output to. If not provided a random filename will be used.
String	No	String	This is the string to use in the Captcha image.

Example

The following example demonstrates creating a Captcha image. The image is displayed and the returned structure is dumped.

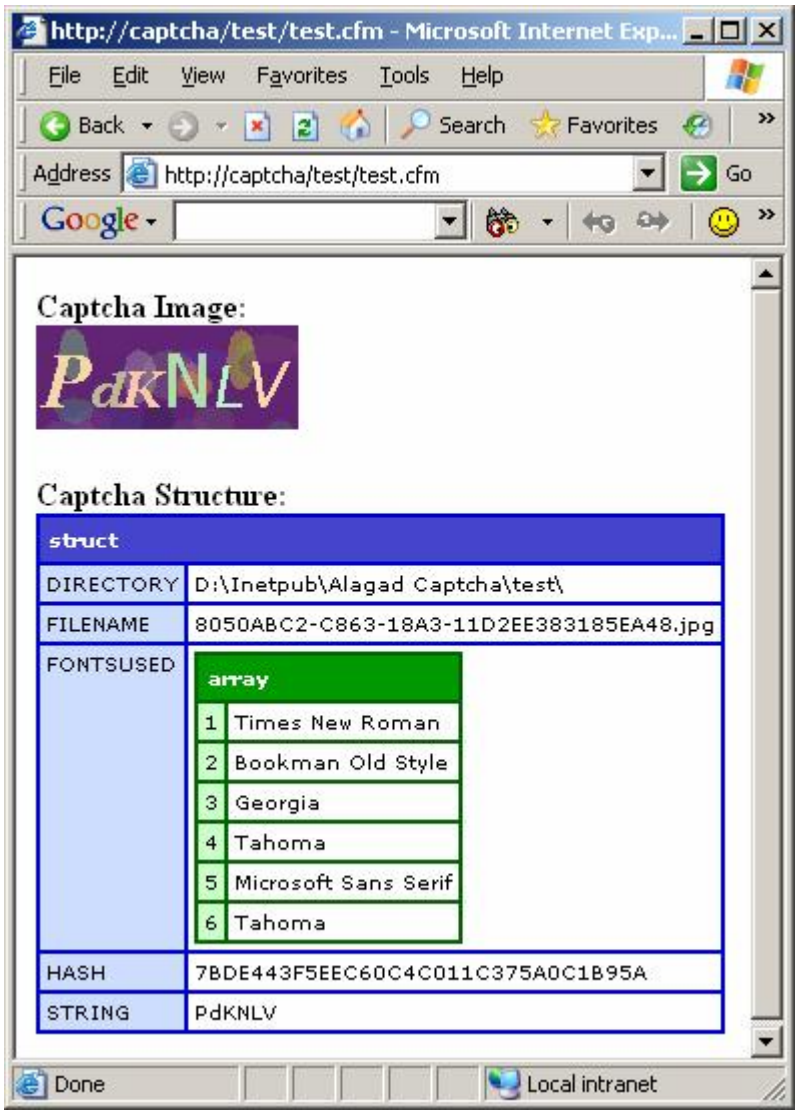
```
<cfset captcha = CreateObject("component", ¬
"Captcha.Captcha").configure(expandPath("."), ¬
"XXXXX-XXXXX-XXXXX-XXXXX-XXXXX")  />

<cfset captchaResults = captcha.createCaptcha() />

<p><b>Captcha Image:<br></b>
<cfoutput>
<img src="#captchaResults.fileName#" />
</cfoutput>
</p>

<p><b>Captcha Structure:<br></b>
<cfdump var="#captchaResults#" />
</p>
```

Results



validate()

Description

The validate() method is used to confirm that the text provided by your users against the hashed version of the string. The hashed version is a lowercase version of the string. The user provided string will be converted to lowercase before comparing.

If this method returns true you can be reasonably confidant that a human provided the string.

Syntax

Boolean = validate(hash, string)

Parameter	Required	Type	Description
Hash	Yes	String	This is the hash of the string displayed in the Captcha image. The string is converted to lowercase before being hashed.
String	Yes	String	This is the string provided by the user. This string is converted to lowercase, hashed and compared to the provided hash attribute

Example

The following snippet of code demonstrates how to call the validate() method.

```
<cfif myCaptcha.validate(session.captchaHash, form.captchaText)>
      <h3>Captcha Is Correct!</h3>
<cfelse>
      <h3>Captcha Is Not Correct!</h3>
</cfif>
```

createRandomString()

Description

The createRandomString() method can be used to create a string of random characters of a specific length. This method excludes certain characters and letters which may be difficult to tell apart. For example 1 (one) and l (lower case L) are not removed.

A good use for this method is to create a longer random string to pass into the string attribute of the createCaptcha() method.

Syntax

String = createRandomString(length)

Parameter	Required	Type	Description
Length	Yes	Numeric	This is the number of random characters to return.

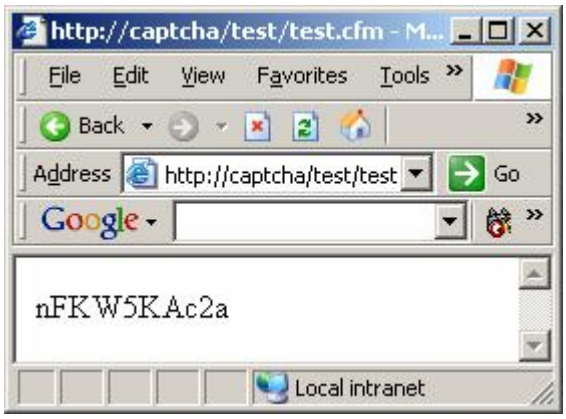
Example

The following example simply generates a random string 10 characters long and outputs it.

```
<cfset captcha = CreateObject("component", ¬
"Captcha.Captcha").configure(expandPath("."), ¬
"XXXXX-XXXXX-XXXXX-XXXXX-XXXXX") />

<cfoutput>#captcha.createRandomString(10)#</cfoutput>
```

Results



setAllowedFontList()

The setAllowedFontList() can be used to specify a list of fonts for the Captcha component to use when generating images. This is of particular importance if you have fonts installed on your system which do not have displayable glyphs for all of the characters which might be displayed by the Captcha component (most systems).

Fonts specified as allowed are the pool from which the Captcha Component will draw when selecting fonts. If no fonts are provided any system font will be used.

This method was introduced to augment and potentially replace usage of the setIgnoredFontlist() method.

Syntax

setAllowedFontList(list)

Parameter	Required	Type	Description
List	Yes	String	This is a comma delimited list of font names which can be used by the Alagad Captcha component.

getAllowedFontList()

Description

The getAllowedFontList() method returns the list of fonts which you have specified as being allowed with the setAllowedFontList() method.

Syntax

List = getAllowedFontList()

setIgnoredFontList()

Description

The setIgnoredFontList() can be used to insure that the Captcha component does not use specific fonts when generating images. This is of particular importance if you have fonts installed on your system which do not have displayable glyphs for all of the characters which might be displayed by the Captcha component (most systems).

If you see an empty square in your Captcha image where a character should be displayed you will want to watch the FontsUsed element in the structure returned from createCaptcha(). The font names in the array correspond to characters in the image as read left to right. You can use this information to create a comma delimited list of font names which should not be used by the component.

This method is deprecated in favor of the setAllowedFontList() method.

Syntax

setIgnoredFontList(list)

Parameter	Required	Type	Description
List	Yes	String	This is a comma delimited list of font names which should not be used by the Alagad Captcha component.

getIgnoredFontList()

Description

The getIgnoredFontList() method returns the list of fonts which you have specified as being ignored with the setIgnoredFontList() method.

Syntax

List = getIgnoredFontList()

setSpotsEnabled()

Description

The setSpotsEnabled() is used to disable or enable the “spots” shown behind a Captcha Image. The purpose of the spots is to confuse OCR software. Passing false to this method will remove the spots from your image. Spots are enabled by default.

Syntax

setSpotsEnabled(spotsEnabled)

Parameter	Required	Type	Description
SpotsEnabled	Yes	Boolean	This argument indicates if the component will draw random spots in the background to confuse OCR software. Options are: True – (default) spots are shown False – spots are not shown

getSpotsEnabled()

Description

The getSpotsEnabled() method returns a Boolean value indicating if "spots" will be drawn into in the Captcha Image to confuse OCR software.

Syntax

SpotsEnabled = getSpotsEnabled()

setContrast()

Description

The setContrast() is adjust the amount of contrast between background and foreground colors. The default value is 0 causes Captcha images be generated with lower contrast to confuse OCR software. You can adjust the amount of contrast by increasing this value. Passing a value of 100 creates a black background with white text on the foreground.

Syntax

setContrast(contrast)

Parameter	Required	Type	Description
Contrast	Yes	Numeric	Used to adjust the amount of contrast between foreground and background colors. Valid values are from 0 (low contrast) to 100 (high contrast / black and white).

getContrast()

Description

The getContrast() method returns a numeric value indicating the current contrast settings.

Syntax

Contrast = getContrast()

setInvert()

Description

The setInvert() method us used to set if the Captcha Component should invert the colors in the image. The default setting of false causes the Captcha Component to create images with light text on a dark background. Testing has shown this to be more effective at confusing OCR software. You can invert the image so that dark text is drawn on a light background by passing true to this method. This may make it easier for some people to read the text of the image.

Syntax

setInvert(Invert)

Parameter	Required	Type	Description
Invert	Yes	Boolean	Indicates if the image should be inverted. False – (default) Indicates the Image will be drawn with light text on a dark background. True – Indicates the Image will be drawn with dark text on a light background.

getInvert()

Description

The getInvert() method returns a Boolean value indicating if the image will be drawn inverted.

Syntax

Invert = getInvert()