



Documentation – Version 0.1 Beta

Last Updated on May 13, 2007

Peter J. Farrell

<http://lyla.maestropublishing.com>

What is a Captcha?

A Captcha is a type of challenge-response test used to determine whether or not the user is human. A common type of captcha requires that the user type the letters of a distorted, sometimes with the addition of obscured sequence of letters or digits that appears in an image. A Captcha can also be administered in the form of question in which the user provides the correct answer or by having the user select similar objects from a set of images (like selecting all the dogs images from a set of animals images).

What does Captcha stand for?

Captcha is an acronym for "completely automated public Turing test to tell computers and humans apart" and was coined in 2000 by a few computer scientist at Carnegie Mellon University and IBM. However, Alan Turing described challenge-response tests in a 1950's paper. While the modern day Captcha was developed at AltaVista in 1997 to prevent bots from adding URLs to the search engine. Everything else is history.

What applications do Captchas have?

Captchas are used to prevent bots from consuming your website's services. Some applications include, but are not limited to preventing bots from:

- registering for free email accounts
- using bulletin boards, listserves, instant messaging services, etc.
- responding to online polls
- filling out online contact forms

Are Captchas accessible?

The simple answer is it depends on implementation. Visual based Captchas are based on the ability to read text which excludes visually impaired users from using protected services because common assistive technologies such as screen readers cannot interpret them.

Some implementation allow users to access audio based Captchas. However, even today these implementations are not as prevalent on the internet and Lyla Captcha only implements visual based Captchas. Despite some accessibility issues, Captchas are

being used by many large internet services like PayPal, Gmail, Hotmail, Yahoo and are commonly seen on weblogs.

How is LylaCaptcha licensed?

LylaCaptcha is licensed under an Apache 2.0 style license and solely owned by the Maestro Publishing, LLC. You can use LylaCaptcha in any commercial application as long as you abide by the license and additional license restrictions.

What are the additional license restrictions?

- The "LylaCaptcha" attribution text must be displayed in the generated Captcha image. Please do not comment out or otherwise change the the source code to defeat this request.
- You may distribute your application in which LylaCaptcha package has been embedded. Please to not sell the LylaCaptcha package by itself as this against the spirit of this open source project.

Why did you name it Lyla?

I could tell you about the reason behind my madness, but I'd rather not. Let's just say it's an homage to a very understanding person in my life. The air of mystery is more fun anyways. Maybe you can figure it out. Clues exist in this documentation.

Who created and maintains LylaCaptcha?

LylaCaptcha was created by Peter J. Farrell and was based on the work of Mark Mandel's original open source ColdFusion Captcha. If you have any questions, comments, bug reports or want to contribute bug fixes or new features, feel free to send a message to Peter J. Farrell at lyla@maestropublishing.com. Please do not feel bad if you don't get an immediate response as I am pretty busy these days and I do not get paid for running this project.

What platforms does LylaCaptcha run on?

LylaCaptcha has been designed to run on Adobe ColdFusion MX6.1 and 7. It has not been tested on New Atlanta BlueDragon, but should work on version 7+ only. Lyla does not run on BlueDragon.NET because it does not have native access to the Java AWT package which is used to render the images.

How do I install LylaCaptcha?

Lyla does not require any ColdFusion mappings or other special installation requirements. You should be able to unzip the package and use it. It has been confirmed that Lyla runs on headless Linux/Unix systems. Please contact me if you encounter any problems on headless systems. You might have to modify a headless Linux/Unix system's JVM configuration file for headless operation.

LylaCaptcha uses a hash reference validation system which requires the service to be stateful. *Meaning you must instantiate (i.e. setting the init'ed CFC) it into a persistent scope like the application or server scope.* Otherwise, the hash reference stored in an LRU type cache will be lost and the Lyla will be unusable.

I'm getting some strange errors when upgrading my Lyla package?

Be sure that you have cleared the template cache in the ColdFusion administrator or restart your ColdFusion server to pick up the new CFCs.

How do I init and setup the Captcha service?

You must call `init()` when using Lyla. You can setup the Captcha service via the XML configuration file (recommended) or you can use populate the `captchaConfigBean` yourself. *You must **init()** into a persistent scope (e.g. server or application scope).*

XML configuration file method (recommended):

```
<cfif NOT StructKeyExists(application, "captcha")>
<cfset application.captcha = CreateObject("component",
"pathTo.captchaService").init(configFile="/pathTo/captcha.xml") />
<cfset captcha.setup() />
</cfif>
```

Prepopulated captchaConfigBean:

```
<cfif NOT StructKeyExists(application, "captcha")>
```

```
<cfset application.captcha = CreateObject("component",  
"path.to.captchaService").init(configBean=myCaptchaConfigBean) />  
<cfset captcha.setup() />  
</cfif>
```

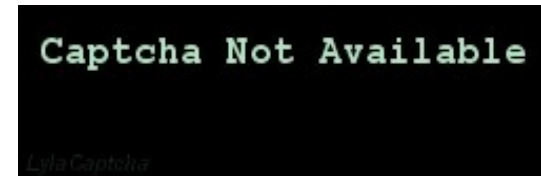
The setup() method may seem silly, this was built-in for users that use dependency injection for their config file or config bean via property injection in ColdSpring . Voila! The service is ready to use. It's is best practice for you to store the Captcha service in a persistent scope (i.e. application) or use ColdSpring to manage your objects. If you re-init'ed the service on every request, Lyla will NOT work and give you strange errors.

Why do I get this error: Object Instantiation Exception. An exception occurred when instantiating a java object. The cause of this exception was that: ?

The outputDirectory setting in your XML config file is incorrect. If you have outputDirectoryIsRelative set to TRUE, you must make sure that your outputDirectory exists. For example, ./img/ points to a directory relative to the calling file. If you use /img/, it points to a directory relative to your web root.

Why do I keep getting a CAPTCHA image that says "Captcha not available"?

You're probably getting something like this image. The font is automatically rotated so example shown to the right will differ somewhat from the image you may get:



You'll get this type of image if you pass in a "" string of text for the CAPTCHA or the hash reference lookup in the cache was unsuccessful. The captchaService.cfc is a stateful CFC because it holds a LRU type cache for hash references. Also, if you have the saltType config option to auto, the salt is automatically generated for that instance of the class. *The captchaService must be init'ed into a persistent scope (i.e. application or server) to function properly.*

How do setup LylaCaptcha with ColdSpring?

Here is a code snip for you to use in your ColdSpring configuration file:

```
<bean id="captchaService"
```

```

class="path.to.captchaService"
init-method="setup">
<constructor-arg name="configFile">
    <value>/path/to/captcha.xml</value>
</constructor-arg>
</bean>

```

How do I use a Hash Reference?

A Hash Reference in Lyla is a hash (one-way encryption) of the original Captcha text and the salt value as defined in the XML configuration file. When you want to validate a user response, you pass the hash and the user's response to `validateCaptcha()` method. The function takes the user's response and the current salt value and check if it matches the hash reference. If the hashes match, the user entered a correct response.

- Create a hash reference using `createHashReference()`.
- Use the hash reference in a IMG tage to call a cfm page the actually generates the Captcha image.

```


<input name="HashReference" type="hidden" value="#variables.captcha.hash#" />

```

- `DisplayCaptcha.cfm` uses the passed `hashReference` url parameter to call `createCaptchaFromHashReference()` which actually creates the Captcha in the desired output (file or stream) and returns the result structure. This cfm page then returns the Captcha using `cfcontent`.
- Once the user completes the activity that the Captcha guards, you then take the user's response and the hash the Hash Reference that was stored in a hidden input and pass then to `validateCaptcha()`. You can then use `cflocation`, `cfthrow` or any custom error handling technique if the validation returns false (meaning the user's response did not match the Captcha text).

Please note the Lyla uses a Least Recently Used (LRU) Cache for Hash References. You must store the Captcha in a persistent scope (i.e. application or server scope) for the service to function properly. All Hash References are automatically expired and cleaned up from the service as a period of time. All Hash References are valid for 30 minutes by default and old hashes are cleaned up from

the LRU during a cleanup cycle.

How do I serve a Captcha using `<cfcontent>`?

To stream:

```
<cfset variables.captcha = application.captcha.createCaptcha("stream") />
<cfcontent type="image/jpg" variable="#variables.captcha.stream#" reset="false" />
```

To file:

```
<cfset variables.captcha = application.captcha.createCaptcha("file") />
<cfcontent type="image/jpg" file="#variables.captcha.fileLocation#" deletefile="true"
reset="false" />
```

How can I test LylaCaptcha?

Run this code in a file called `captchaTest.cfm` that is location in the same directory as the LylaCaptcha components:

```
<cfif NOT StructKeyExists(application, "captcha")>
<cfset application.captcha = CreateObject("component",
"captchaService").init(configFile="captcha.xml") />
<cfset application.captcha.setup() />
</cfif>

<cfset variables.hashReference = application.captcha.createHashReference() />

<cfif StructKeyExists(url, "type") AND url.type EQ "stream">
    <cfset variables.captcha =
application.captcha.captchaFromHashReference(variables.hashReference.hash, "stream") />
    <cfcontent type="image/jpg" variable="#variables.captcha.stream#" reset="false" />
<cfelseif StructKeyExists(url, "type") AND url.type EQ "file">
    <cfset variables.captcha = captchaFromHashReference(variables.hashReference.hash,
```

```
"file") />
    <cfcontent type="image/jpg" file="#variables.captcha.fileLocation#" deletefile="true"
reset="false" />
<cfelse>
    <p>Please select a Captcha stream type:</p>
    <ul>
        <li><a href="captchaTest.cfm?type=stream">Stream</a> (Only works on CFMX7+)</li>
        <li><a href="captchaTest.cfm?type=file">File</a></li>
    </ul>
</cfif>
```

✿Configuring the Rendering Engine

How do I configure LylaCaptcha?

The easiest (and most recommended way) to configure LylaCaptcha is to use the XML configuration file. The configuration file consists of 33 config type directives as well as the ability to define the fonts to use in your Captchas. Lyla ships with a decent rendering settings. If you intend to create Captchas to a file, you should just have to set the `outputDirectory` setting and be able to use Lyla.

Config Name	Default	Type	Accepted Values	Description
<code>outputDirectory</code>	<code>img/</code>	string	valid path or mapping	Directory path to store generated Captchas
<code>outputDirectoryIsRelative</code>	<code>true</code>	boolean	<ul style="list-style-type: none">• true• false	Switch is the output directory is relative [Uses <code>ExpandPath()</code>]
<code>hashValidityPeriod</code>	<code>auto</code>	numeric	0 - ?	The length of time in milliseconds that the hash will be valid for.
<code>jpegQuality</code>	<code>0.90</code>	numeric	decimal between 0-1.00	The quality setting for JPEG.
<code>jpegUseBaseline</code>	<code>true</code>	boolean	<ul style="list-style-type: none">• true• false	Switch for JPEG baseline.
<code>useAntiAlias</code>	<code>true</code>	boolean	<ul style="list-style-type: none">• true• false	Switch for JPEG anti-aliasing.

randStrType	alpha	string	<ul style="list-style-type: none"> • alpha • alphaLCase • alphaUCase • alphaNum • alphaNumLCase • secure (alphaNum + special) 	Type of random string to generate for the Captcha unless the text is passed into the service
randStrLen	6	numeric	whole number only	*Average number of characters to use in the Captcha (varies -1 to +1 of the value)
width	250	numeric	whole number only	Desired width
height	75	numeric	whole number only	Desired height
fontsize	30	numeric	whole number only	*Average font size to use in the Captcha
leftOffset	20	numeric	whole number only	*Average number of pixels to offset the first character
shearXRange	25	numeric	whole number only	*Average X axis shear range
shearYRange	25	numeric	whole number only	*Average Y axis shear range
fontColor	light	string	<ul style="list-style-type: none"> • light • medium • dark • lightGray • mediumGray • darkGray 	The color range to use for the Captcha characters

backgroundColor	dark	string	<ul style="list-style-type: none"> • light • medium • dark • lightGray • mediumGray • darkGray 	The color range to use for the background
useGradientBackground	true	boolean	<ul style="list-style-type: none"> • true • false 	Switch for gradient type background
backgroundColorUseCyclic	true	boolean	<ul style="list-style-type: none"> • true • false 	Switch for wavy/oscillating background.
useOvals	true	boolean	<ul style="list-style-type: none"> • true • false 	Switch for ovals in the background
ovalColor	medium	string	<ul style="list-style-type: none"> • light • medium • dark • lightGray • mediumGray • darkGray 	The color range to use for ovals
ovalUseTransparency	true	boolean	<ul style="list-style-type: none"> • true • false 	Switch for transparency (alpha channel) when creating ovals
minOvals	15	numeric	whole number only	Minimum number of ovals
maxOvals	20	numeric	whole number only	Maximum number of ovals
useBackgroundLines	true	boolean		Switch for background lines

backgroundLineColor	medium	string	<ul style="list-style-type: none"> • light • medium • dark • lightGray • mediumGray • darkGray 	The color range to use for background lines.
backgroundLineUseTransparency	true	boolean	<ul style="list-style-type: none"> • true • false 	Switch for transparency (alpha channel) when creating background lines
backgroundMinLines	5	numeric	whole number only	Minimum number of background lines
backgroundMaxLines	10	numeric	whole number only	Maximum number of background lines
useForegroundLines	true	boolean	<ul style="list-style-type: none"> • true • false 	Switch for foreground lines
foregroundlineColor	light	string	<ul style="list-style-type: none"> • light • medium • dark • lightGray • mediumGray • darkGray 	The color range to use for foreground lines.
foregroundLineUseTransparency	true	boolean	<ul style="list-style-type: none"> • true • false 	Switch for transparency (alpha channel) when creating foreground lines

foregroundMinLines	5	numeric	whole number only	Minimum number of foreground lines
foregroundMaxLines	10	numeric	whole number only	Maximum number of foreground lines

How do I know which fonts to use?

Lyla provides a utility method called `getAvailableFontNames()` to help you decide what fonts are available on the host system. All you need to do is select the fonts you want to use in the XML configuration file. Example:

```
<font>
  <font use="true" name="Arial"/>
</font>
```

What are the public methods?

Other than the public `init()` and `setup()` methods that used for setting up the service, there are 3 general methods and 2 utility methods.

Method Name	Type	Return Type	Results	Description
<code>createHashReference()</code>	general	struct	<ul style="list-style-type: none"> • type = hash • text = random string • hash = hash reference • width = width of image • height = height of image 	Creates a hash reference in the service. Does not actually create a Captcha, but creates a reference so a Captcha can be created in a future request.

createCaptchaFromHashReference ()	general	struct	<ul style="list-style-type: none"> • type = hash • text = random string • hash = hash reference • width = width of image • height = height of image <p>If argument type is “file”:</p> <ul style="list-style-type: none"> • fileLocation (combination of both fileDirecotry and fileName) • fileDirectory • fileName <p>If argument type is “stream”:</p> <ul style="list-style-type: none"> • stream (Java byte array of image to use with cfcontent) 	Creates a captcha to the desired from a hash reference.
validateCaptcha ()	general	boolean	<ul style="list-style-type: none"> • true • false 	Validates a captcha by hash and user response text.
getAvailableFontNames ()	utility	array		Returns an array of all available system fonts. This is useful when deciding on fonts to use for Captcha configuration.

<code>getVersion()</code>	utility	string	Returns the current version number of LylaCaptcha.
---------------------------	---------	--------	--

What does Lyla's default Captcha rendering looks like?



Is LylaCaptcha safe for use on sticky session clustered servers?

Yes. However, Lyla does not run on non sticky session clustered servers since Lyla manages an LRU cache which manages the hash references.

What file formats can LylaCaptcha create?

Currently, LylaCaptcha can only create JPEGs. It cannot currently create PNGs because Java does not ship with an encoder for that file format. Support for PNGs is planned, but PNG file sizes tend to be larger than JPEGs and I have yet to receive a comment that sways me on that point. I have no plans to support GIFs due to problems with patents.

How many KBs are the files that Lyla creates?

At the default 0.90 JPEG quality, most Captchas are 6-10KBs with the config settings that ship with Lyla. Of course, file size depends on the JPEG quality setting and the size of the captcha.

What about your plans for future versions of LylaCaptcha?

Future plans for Lyla is to implement a background factory and distortion engine.

Who uses LylaCaptcha?

Websites:

- GreatBizTools (<http://www.greatbiztools.com>)
A human resource consulting company that provides “Innovative Solutions for Managing Your Business”.

ColdFusion applications:

- MachBlog (<http://www.machblog.org>)
An open source weblog application built on Mach-II and ColdSpring.
- BoardFusion (<http://www.boardfusion.org>)
An open source bulletin board application for ColdFusion.
- BlogCFC – Version 5.0 (<http://www.blogcfc.com>)
An open source weblog application by Raymond Camden.
- BlogFusion – (<http://www.blogfusion.com>)
A ColdFusion blogging community.

Let me know if you use LylaCaptcha and want to be added to the list. Please include the website address, use type (website or application) and a brief description.

Who does the author want to thank?

My first thank you goes to Mark Mandel from Compound Theory (<http://www.compoundtheory.com>) for releasing the base code that inspired Lyla Captcha’s first working incantations. Thank you Mark for all your hard work and your blessings to open source this derivative work.

Secondly, I would like to thank two cflib.org authors for their UDFs which made it into the LylaCaptcha package. A big thanks to Rob Brooks-Bilson for SHA1() and Nathan Dintenfass for ArrayFind().

Thirdly, I’d like to thank my one of my employers – GreatBizTools (<http://www.greatbiztools.com>) for allowing me to open source LylaCaptcha since I originally wrote the base version for them. My best goes to Denise and Barry for being so supportive of the open source software movement.

Lastly, I’d like to thank dearest Ally– my partner in my life of “crime” and the light of my life. Thanks honey for letting me stay up

late to work on the “non-work” as it is sometimes called here at home.

How can I show my appreciate to the author or supporting the author’s Shameless Begging?
If LylaCaptcha proves to be a useful Captcha for you, please consider gifting me something on my Amazon Wishlist:

<http://www.amazon.com/gp/registry/2NDKJIPDUYE9W>

Or you may make a donation with PayPal:

pjf@maestropublishing.com

Your generosity is greatly appreciated and thank you for supporting Open Source Software authors.