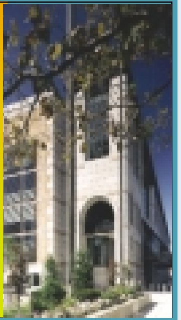
SPECIAL ISSUE: COLDFUSION SECURITY

ColdFusion Developer's Journal

ColdFusionJournal.com

September 2004 Volume: 6 Issue: 9

web services EDGE conference & expo
Web Services Edge 2005 East
February 15-17, 2005
Hynes Convention Center
Boston, MA
See page 39 for details



RETAILERS PLEASE DISPLAY UNTIL NOVEMBER 30, 2004

$9.99US $9.99CAN

SYS-CON MEDIA

By Alex Hübner page 8

MOBILE INC. Telecommuncations for the modern age
Home
News
Calendar
Discussion
Library
Directory
Mobile Inc. announc
sleek new model
FOR IMMEDIATE RELEASE
SAN FRANCISCO – Mobile Inc. is revolutionizi
raising the bar and delivering one of the most a
phones to date. This ergonomically designed
will be targeted to business leaders on the go
mobile phone that travels lightly and conven

When was the last time your Intranet was updated?

Announcing the Macromedia® Web Publishing System. You build and manage the site. Users keep content fresh.

With Macromedia® Studio MX 2004, Contribute™ 3 and Contribute Publishing Services, you can affordably build, manage, and publish enterprise sites. And unlike the average content management system, this works. You determine who can edit and who can publish. Then users simply point and click to update any page. In no time, dated content will be a thing of the past. Learn more at **www.webpublishingsystem.com**

macromedia®
WEB PUBLISHING
SYSTEM

SUPERIOR MANAGED HOSTING

- Intelligent Routing
- Redundancy
- Network Security
- Service Level Agreement
- Environmental Controls
- Network Uptime Guarantee
- Scalability
- OC3/SONET Backbone
- Backup Power
- Physical Security
- Fire Protection
- Server Hardware Guarantee

IF YOU'VE BEEN SEARCHING for a reliable managed hosting partner, your search for greener pastures is over. There is no longer a need to settle for inferior support and lack of accountability. ServerSide takes the guess work, and the associated hassles, out of working with a technology partner.

ServerSide provides managed web hosting solutions for a wide range of customers including; Fortune 500 corporations, small and medium sized businesses and non-profit organizations located across the United States and around the World.

We provide a single point of accountability for all your web hosting infrastructure needs — while adding value, not cost.

>> SPECIAL OFFER

Sign up online for a new shared hosting account now at www.serverside.net and ServerSide will waive the set-up fee.

ENTER CODE: mxd2004

The grass is greener at
www.serverside.net
888.682.2544
hosting@serverside.net

serverside™
Dynamic Web Development. Superior Managed Hosting.

It’s everybody’s PDF™

Finally, a software company that offers affordable yet flexible PDF solutions to meet every customer's needs. Using activePDF™ to automate the PDF creation process eliminates the need for end-user intervention so your employees can concentrate on what they do best.

Licensed per server, activePDF solutions include a COM interface for easy integration with ColdFusion. Dynamically populate PDF forms with information from a database, convert ColdFusion web pages to PDF on the fly, dynamically print reports to PDF using CF and much more. Users can also merge, stitch, stamp, secure and linearize PDF, all at a fraction of the cost of comparable solutions. Download your free trial version today!

activePDF®
Leading the iPaper Revolution

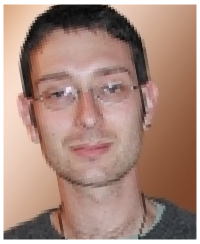
www.activePDF.com

editorial offices

SYS-CON MEDIA
135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 **FAX:** 201 782-9638
COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
is published monthly (12 times a year)
by **SYS-CON Publications, Inc.**
postmaster: send address changes to:
COLDFUSION DEVELOPER'S JOURNAL
SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645

editorial

Reaching Deeper



By Simon Horwith

Now that I've begun settling in to the role of editor-in-chief of ***CFDJ***, I'm beginning to incorporate some ideas I have for the magazine. The first change you'll note this month is that **Tales from the List** is gone. That's not because I'm too lazy to both edit the magazine and write a monthly column – far from it in fact. Though the column is meant to focus on threads from the *CFDJ-List*, as the title implies, I have also used it from time to time as a forum to discuss community events and trends. I think that's a good thing, and so the column is now titled **Community Focus**. Each month, the column will discuss one or more community events, trends, discussion list threads, blog entries, Macromedia announcements, or other community-related topics.

Also new this month is a bit of an experiment – a pretty heavy focus on specific topics, including articles with similar if not identical themes written by different authors. I want to see what happens if we choose a few specific topics and explore them more deeply, rather than running a mixed bag. This month, *CFDJ* has two focus topics: security and "selling" ColdFusion.

On security, Wayne Graham has written a terrific article about a rather advanced topic – how to use public-key encryption in ColdFusion applications. The nice part is that he uses nothing but free Java-based software in his examples so no developer working with sensitive data has an excuse to not be using public key–based encryption/decryption. Alex Hubner's article introduces and explains ColdFusion's Sandbox Security. If you've never explored the security features built into ColdFusion, I strongly suggest you read his article. Michael Smith of TeraTech and CFUN fame and security expert Bryan Murphy both list their top tips for building secure applications. Even my community focus article is getting in on the action, as I recap several recent security-related blog entries.

Another focus topic this month is "how to sell ColdFusion." It seems like this has been a hot topic for ColdFusion developers for longer than I can remember – more so now than ever before. We have two ColdFusion legends giving their take on the subject. Ben Forta has written an article based on several recent blog entries and e-mail exchanges that summarize ColdFusion "selling points" that are specific to the alternative technology being evaluated. Hal Helms' article is based on a discussion he recently had with a client who had to justify the use of ColdFusion as a development platform. Both articles are a must-read, as we all end up in the same situation of defending CF to a client sooner or later. Be prepared!

Also this month is an interesting article by Harry Klein about a Mach-II unit testing framework, a jointly authored article by Matthew Fusfield and Ryan Emerle about how they used Java to monitor networks, and Jeffry Houser's monthly **CF101** column in which he discusses the ColdFusion application framework.

Please let me know what you think of the "deeper reach" format we've experimented with this month by dropping me an e-mail at simon@horwith.com If enough of you approve, I'll do my best to ensure we produce more issues like it. If not: back to the drawing board! If you have any suggestion(s) for articles, regular columns, other changes to the content, or any other ideas for *CFDJ*, now's the time to speak up. In the meantime, I have a few of my own experiments planned. Your feedback is invaluable in determining what does and does not work, so please do let me know.

About the Author

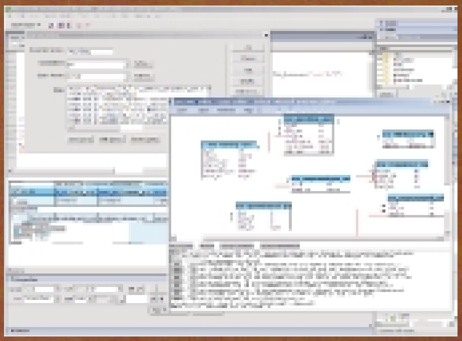
Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal. *He is a freelance software architect currently consulting with companies in London, England. Simon is a Macromedia Certified Master Instructor and is a member of Team Macromedia. He has been using ColdFusion since version 1.5 and consults on ColdFusion applications that leverage Java, Flash, Flex, and a myriad of other technologies. In addition to presenting at CFUGs and conferences around the world, he has also been a contributing author of several books and technical papers. You can read his blog at www.horwith.com*
simon@horwith.com

Need a hand with your work? Get the MX Kollection.

Introducing MX Kollection - the suite of extensions designed to change the way you create dynamic web applications with Dreamweaver MX and MX 2004.

Our customers think of MX Kollection as the next level in Dreamweaver MX visual software development.

ColdFusion and PHP developers will find our product enabling them to visually develop e-Commerce, CMS, CRM, Backend and other web solutions with increased efficiency, quality and capability.



Create powerful dynamic lists

- Sort, filter and page result sets
- Perform multiple deletes
- Easily create Master/Detail lists

Build unified add/modify forms

- Client side and server side validation
- Insert into two tables
- Create form actions such as send mail or delete related record
- File/Image upload and resize

Create recordsets visually

- Build complex SQL queries across multiple tables quickly

... and many more

Interakt

macromedia ALLIANCE PARTNER

Download the demo and see the features and benefits of our extensions at http://www.interaktonline.com

SYS-CON MEDIA

president & ceo
Fuat Kircaali fuat@sys-con.com

vp, business development
Grisha Davida grisha@sys-con.com
group publisher
Jeremy Geelan jeremy@sys-con.com

advertising
senior vp, sales & marketing
Carmen Gonzalez carmen@sys-con.com
vp, sales & marketing
Miles Silverman miles@sys-con.com
advertising director
Robyn Forma robyn@sys-con.com
advertising manager
Megan Mussa megan@sys-con.com
associate sales managers
Kristin Kuhnle kristin@sys-con.com
Beth Jones beth@sys-con.com
Dorothy Gil dorothy@sys-con.com

sys-con events
president, events
Grisha Davida grisha@sys-con.com
National Sales Manager
Jim Hanchrow jimh@sys-con.com

customer relations
circulation service coordinators
Edna Earle Russell edna@sys-con.com
Linda Lipton linda@sys-con.com
Monique Floyd monique@sys-con.com

manager, jdj store
Brunilda Staropoli bruni@sys-con.com

sys-con.com
vp, information systems
Robert Diamond robert@sys-con.com

web designers
Stephen Kilmurray stephen@sys-con.com
Matthew Pollotta matthew@sys-con.com

online editor
Martin Wezdecki martin@sys-con.com

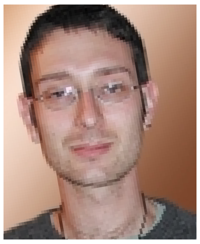
accounting
financial analyst
Joan LaRose joan@sys-con.com
accounts payable
Betty White betty@sys-con.com
accounts receivable
Shannon Rymsza shannon@sys-con.com

subscriptions
Subscribe@sys-con.com
1 888 303-5282
201 802-3012
For subscriptions and requests for bulk orders, please send your letters to Subscription Department
Cover Price: $8.99/issue
Domestic: $89.99/yr (12 issues)
Canada/Mexico: $99.99/yr
All other countries $129.99/yr
(U.S. Banks or Money Orders)
Back issues: $12 U.S. $15 all others

Problems Securing Your Applications?

Advice from the experts

This marks the first installment of a new monthly column, **Community Focus**, and the end of a long-running column, **Tales from the List**.



By Simon Horwith

Due to the rapid growth of ColdFusion community news and events and the growing popularity of blogs (Web logs) amongst ColdFusion developers, it no longer seems appropriate to focus on one list server for a "messages from the trenches" type article. Each month, **Community Focus** will explore trends, Macromedia news, events, blog entries, and list server threads, focusing on what's going on in the community… or at least one facet of what's going on in the community. This month's issue is focused heavily on security, so following a series of blog entries, I will join suit and do the same.

This month's column is about securing applications or, rather, problems with securing applications. Macromedia recently released a TechNote written by Sarge (www.macromedia.com/support/coldfusion/ts/documents/loginstorage_caching.htm) in which he describes a problem with the CFLOGIN tag. Apparently, when the CFAPPLICATION tag "loginstorage" attribute is set to "session" the CFLOGOUT tag doesn't always properly delete the internal variables used to store session information, only the session variables. This can result in one user logging out and another user logging in on the same browser only to receive the other user's credentials.

Ray Camden blogged about it (Ray's blog can be found at www.camdenfamily.com/morpheus/blog/) and offered a solution that basically consists of doing a CFLOGOUT whenever "session.user" is not defined. This works fine if you don't assume that all users are logged in, but is a bit of a hack in that sometimes you want to assume authentication and just make people who haven't submitted a login form seem as authenticated as an "anonymous" or "guest" user.

For those of you who are not aware, JRun and many/most other Java applications use a standard Java package called JAAS (Java Authentication and Authorization Service) to handle security. ColdFusion, under the hood, also uses this framework to handle authentication/authorization whenever you use the CFLOGIN/CFLOGOUT/CFLOGINUSER tags or getAuthUser() function. I recently blogged about problems with authentication against the JAAS framework in a clustered environment. As many of you may already know, I am a freelance consultant currently working on a contract in London. My current client offers several products to clients and has a single common authentication/authorization module. All of their software offerings are developed in ColdFusion, Java, or a hybrid of the two. We also recommend running our software in a clustered environment. To our dismay, we've found that JAAS authentication on one server in a JRun cluster does not fail-over to other servers in the cluster.

— continued on page 14

About the Author

Simon Horwith is the editor-in-chief of ColdFusion Developer's Journal. *He is a freelance software architect currently consulting with companies in London, England. Simon is a Macromedia Certified Master Instructor and is a member of Team Macromedia. He has been using ColdFusion since version 1.5 and consults on ColdFusion applications that leverage Java, Flash, Flex, and a myriad of other technologies. In addition to presenting at CFUGs and conferences around the world, he has also been a contributing author of several books and technical papers. You can read his blog at www.horwith.com*
simon@horwith.com

Sandbox Security

ColdFusion MX security for shared environments

By Alex Hübner

Despite what your parents told you, sharing is not always a good thing. However, if you provide a shared hosting environment, here are some tips on making it more secure.

Frequently, I come across hosting providers or companies offering ColdFusion on their servers. Unfortunately, many of these companies are not thoroughly familiar with the platform, especially in a shared hosting environment. ColdFusion is a product that is easy to install and manage, but that doesn't mean you should forego reading the documentation provided with the product.

This article does not to attempt to provide a full-fledged guide to securing a ColdFusion server, nor does it examine in depth the technical details of security sandboxes and how they work. What I will discuss here is a small collection of settings that I recommend for shared host servers running ColdFusion Enterprise with sandbox security. Again, this article should not be viewed as a replacement for the product's official documentation. This is a mini how-to, and will not necessarily be adequate for your servers. Use the information provided in accordance with your judgment and experience.

This article covers specifically Macromedia ColdFusion Server versions MX 6.0 and 6.1 Enterprise stand-alone installations. All examples are based on the Windows platform but can be applied to *nix systems if you adapt the information provided (e.g., file system and drive names).

Shared Hosting Environments

A point that needs to be clear from the beginning: shared hosting is never 100% secure. The fact is that when you share server resources you're also sharing inherent risks. In a shared environment, it's not possible to isolate users and applications from each other safely. Any application can interfere with another one and, consequently, the entire server. A classic example is the DoS attack (not to be confused with a DDoS – distributed denial-of-service attack), intentional or not. In fact, many CFML beginners create true "script monsters" that cause a lot of trouble for CPU and memory, compromising the performance of the entire server.

If you insist on an absolutely 100% secure and functional ColdFusion environment, you should go for a dedicated or semi-dedicated solution, never a shared one. But thanks to the underlying Java technology, ColdFusion Enterprise atop a J2EE server (JRun, WebLogic, WebSphere, etc.) – formerly known as "ColdFusion for J2EE" – offers the ability to run multiple isolated instances of your applications/sites in a single machine. This means isolation in terms of security as well as performance (but limited to the hardware resources available on the machine) since each instance has its own JVM.

For those of you unfamiliar with Java, the term JVM stands for Java Virtual Machine, which exists when the software creates its own abstraction, in rough mode, working as an exclusive and separate machine. The JVM approach causes an increased cost both operationally as well as for the computer's resource consumption and it's not the focus of this article. You can find a nice primer at www.macromedia.com/devnet/mx/coldfusion/j2ee/articles/multiple.html and www.macromedia.com/devnet/mx/coldfusion/articles/multi_instances.html.

The Two Commercial Versions of Macromedia ColdFusion

As you might know, there are two versions of the Macromedia ColdFusion Servers available. The first is called Standard (formerly known as Professional) and the second is called Enterprise.

What are the differences? From the programmer's standpoint, there's no difference at all. The support for CFML is almost identical; you'll find some little differences, but they are not significant for most applications. However, for those who host and manage Web applications/sites, the differences are very important and significant. ColdFusion Enterprise offers support to a larger number of platforms and databases. Additionally, the Enterprise version supports the installation of multiple instances (as mentioned above), security sandboxes (indispensable for shared hosting), and many other improvements for the Enterprise scenario. It also includes a complete and fully functional version of the excellent Macromedia J2EE Server: JRun 4.0. A table comparing the two versions can be found at: www.macromedia.com/software/coldfusion/productinfo/product_editions.

As a rule of thumb, if you are offering ColdFusion in a shared environment, your choice must be for the Enterprise version. Unfortunately, many system administrators, unaware (or aware) of the differences, choose to buy Standard, the "cheaper version." However, today's "cheaper version" may end up costing you more in the long run: a shared host with the Standard version is risky – and you are placing your clients and your reputation in danger.

CFML developers usually have good intentions and do not waste time with invasion attempts or other illegal virtual activities within their server. In addition, to explore the vulnerabilities using CFML scripts, it is necessary to have an account in the server (in most cases), which obligates the "hacker" (or just a really bad programmer) to be a client/developer. But, unfortunately, a good system administrator can't rely on the good faith of his or her clients (or their good programming practices), so you'll need security sandboxes.

What Are Security Sandboxes?

If you have cats at home (particularly in an apartment), you know that they need (and like) to use a litter box for their needs. The concept of sandboxes in ColdFusion can be explained in a coarse manner, with an analogy to cats' litter boxes. In these isolated boxes, cats (or users) can do whatever they wish without dirtying their surroundings. What does this mean in practical terms? It means that sensitive tags such as CFFILE, CFDIRECTORY, or others that potentially could cause damage, can be used without problems or worries. The user locked in a well-configured sandbox will not have permission to change or save files in a neighbor account or any other place. The user will not be able to upload, delete, copy, or create files and/or folders outside his/her sandbox. There are, however, some exceptions, which I will discuss later in this article.

It's clear that in a shared host environment you must use sandboxes. Is this already enabled in your server? No? Then, let's do it:

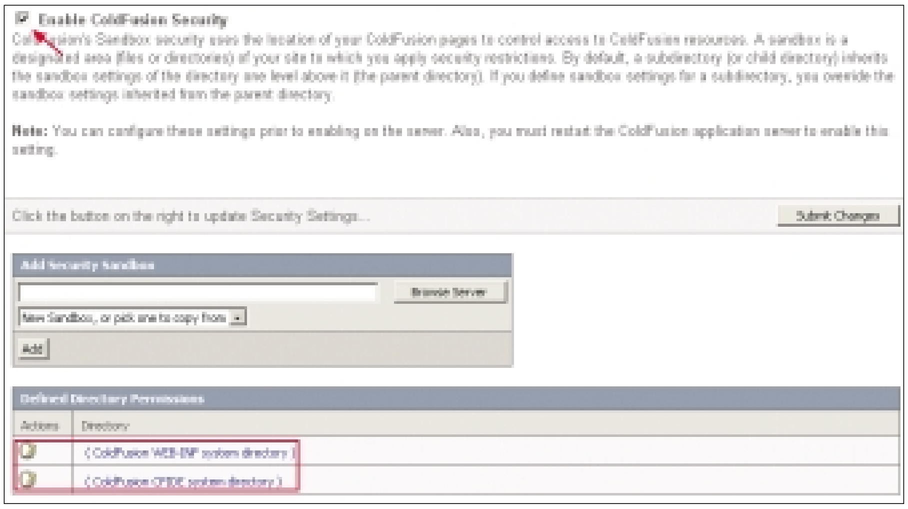
1. Enter in the ColdFusion Administrator.
2. In the left menu, find the option called "Sandbox security".
3. The "Enable ColdFusion Security" option should be disabled. Enable it and click on "submit changes".
4. Restart the ColdFusion Server's service.

Once you have completed the above steps, you should log back in to the ColdFusion Administrator again and check if it has two default sandboxes (see Figure 1).

The two sandboxes shown in Figure 1 (ColdFusion CFIDE and ColdFusion WEB-INF) are necessary for ColdFusion to function properly when using sandbox security. For that reason they may not be removed (note the absence of the delete icon). Make sure to check if the sandbox entry that points to the CFIDE directory corresponds to the directory where the application files for the "ColdFusion Administrator" are stored and used. If the CFIDE folder is in its default installation place, the CFMX should configure it correctly. However, if you have changed the CFIDE folder's location, it will be necessary to change the sandbox, pointing it to your CFIDE folder's location.

Creating New Sandboxes

The process of creating new sandboxes is quite simple. A sandbox is always based in a root folder, usually the user's root folder on the server. To exemplify this, imagine the following scenario: a user account "alexhubner" has a Web site in your server whose root folder was configured in "d:\sites\alexhubner". As expected in shared environment, we will find other user folders below "d:\sites\", such as


☑ Enable ColdFusion Security

ColdFusion's Sandbox security uses the location of your ColdFusion pages to control access to ColdFusion resources. A sandbox is a designated area (files or directories) of your site to which you apply security restrictions. By default, a subdirectory (or child directory) inherits the sandbox settings of the directory one level above it (the parent directory). If you define sandbox settings for a subdirectory, you override the sandbox settings inherited from the parent directory.

Note: You can configure these settings prior to enabling on the server. Also, you must restart the ColdFusion application server to enable this setting.

Click the button on the right to update Security Settings...

Defined Directory Permissions

Actions	Directory
	[ColdFusion WEB-INF system directory]
	[ColdFusion CFIDE system directory]

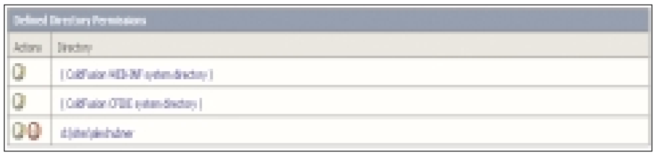
Figure 1: Checking the sandboxes

"d:\sites\joe", "d:\sites\marie", and so forth. The user "alexhubner" should have complete control (read, write, execute, and delete) of files and folders in his root folder "d:\sites\alexhubner" (and below it).

Using this scenario as an example, the creation process consists of the following two easy steps:

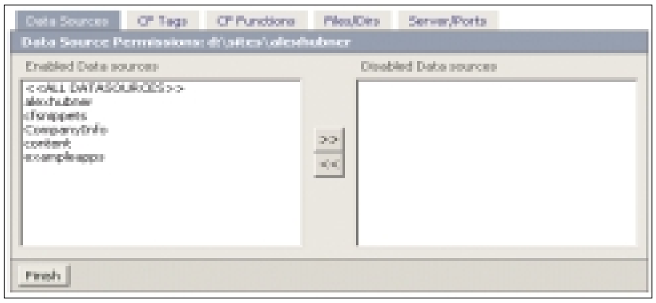
1. In "Add Security Sandbox", specify the location of the user's root folder, in this case "d:\sites\alexhubner" (you can use the browser button to locate the folder).
2. Click on the "Add" button.

When done, a new sandbox, represented by an entry showing the root folder path, should appear in the listing of existing sandboxes (see Figure 2).


Defined Directory Permissions

Actions	Directory
	[ColdFusion WEB-INF system directory]
	[ColdFusion CFIDE system directory]
	d:\sites\alexhubner

Figure 2: Checking the sandboxes

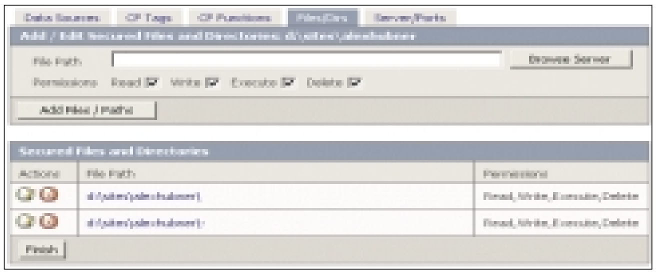


Data Source Permissions: d:\sites\alexhubner

Enabled Data sources:
<<ALL DATASOURCES>>
alexhubner
cfsnippets
CompanyInfo
content
exampleapps

Disabled Data sources

Figure 3: Configuring the new sandbox



Add / Edit Secured Files and Directories: d:\sites\alexhubner

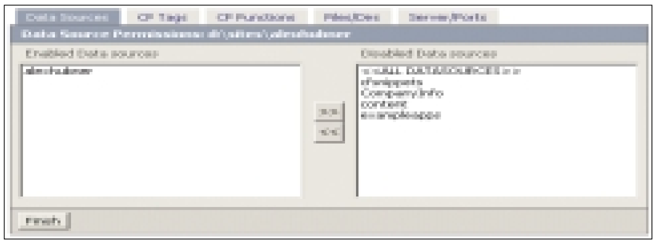
File Path

Permissions Read ✓ Write ✓ Execute ✓ Delete ✓

Secured Files and Directories

Actions	File Path	Permissions
	d:\sites\alexhubner\	Read,Write,Execute,Delete
	d:\sites\alexhubner\-	Read,Write,Execute,Delete

Figure 4: The "Files/Dir" tab



Data Source Permissions: d:\sites\alexhubner

Enabled Data sources:
alexhubner

Disabled Data sources:
<<ALL DATASOURCES>>
cfsnippets
CompanyInfo
content
exampleapps

Figure 5: Configuring the Data Sources tab

The next step is to configure the sandbox settings. The settings I propose are focused on building a more secure ColdFusion box in shared environment.

Configuring Sandboxes for Shared Environments

When clicking on the sandbox we've just created, we'll find an interface with five tabs: "Data Sources", "CF Tags", "CF Functions", "Files/Dirs", and "Server/Ports" (see Figure 3).

The functioning of this interface is quite easy. For all tabs, the left window shows what is enabled and the right window shows what is disabled. By default, upon its creation, each new sandbox will have access to all servers' data sources, tags, functions, and external resources such as Web services and other server's services (HTTP, FTP, etc.). The exception is the manipulation of files and folders outside the user's root folder, "d:\sites\alexhubner" (see Figure 4).

Note: I usually add a third entry (not obligatory) pointing it to the ColdFusion temporary folder. Some commercial applications and custom tags make use of this folder through the function GetTempDirectory(). This function is normally used to store temporary files. The ColdFusion temporary file folder is usually located at C:\CFusionMX\runtime\servers\default\SERVER-INF\temp\wwwroot-tmp\.

We will now take a tour through the tabs and I will share my recommended configurations.

The Data Sources Tab

In the Data Sources tab, we will configure which data sources are allowed to be used by a specific sandbox/account. Imagine that each client has only one data source for his or her hosting plan. In this example the tab should look as shown in Figure 5.

The CF Tags and CF Functions Tabs

I mentioned earlier that there are some exceptions to making a sandbox completely secure. Because of these exceptions (which I explain later), I recommend that some CFML tags and functions be disabled in shared environments.

In the CF Tags tab, I recommend disabling the following tags:

- ***CFExecute:*** This allows software execution at the server level. A good example is the WinZip command-line extension program that is used to zip files. There are many others, such as ping.exe, tracert.exe, etc. In many cases ColdFusion is installed under a privileged account, the "System" account in Windows (this is the default). This is risky because a user can perform an upload of a harmful executable in its base folder and execute it using "System" privileges. Additionally, you can have performance problems if the user uploads and runs a problematic executable that can over-consume the server's resources, causing DoS problems;
- ***CFObject:*** See details in the next section.
- ***CFObjectCache:*** See details in the next section.
- ***CFRegistry:*** It's hard to find a CFML application that needs to access the registry. When one does, it's probably a complex application that integrates with other server's resources and applications, thereby being more suitable

CFDynamics

A Division of Konnections Inc.

RUN YOUR CFML:

Within BlueDragon Server, or as a native J2EE web application.

NOW AVAILABLE!

BlueDragon [6.1]

New Atlanta COMMUNICATIONS

As one of the ColdFusion hosting community leaders, CFDynamics is constantly looking for ways to deliver new and cutting edge technologies to the ColdFusion community. We are excited to announce our premiere partnership with New Atlanta by offering BlueDragon hosting. We look forward to seeing how BlueDragon expands the possibilities of ColdFusion.

WWW.CFDYNAMICS.COM
866 - CFDYNAMICS
866-233-9626

QUALITY CF HOSTING PLANS

- FREE SQL server access
- FREE account setup
- UNLIMITED email accounts
- GENEROUS disk space / data transfer
- 30 day MONEY-BACK GUARANTEE
- GREAT VALUE!

RELIABLE NETWORK

- 99.99% average uptime!
- State-of-the-art data center has complete redundancy in power, HVAC, fire suppression, bandwidth, and security
- 24 / 7 network monitoring

FANTASTIC SUPPORT SERVICES

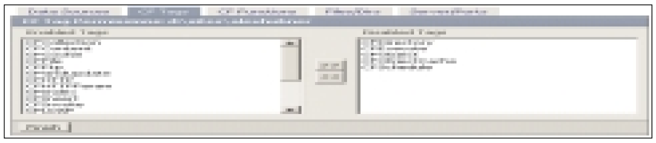
- Comprehensive online support
- Knowledgeable phone support
- We focus on your individual needs

SIGN UP FOR A HOSTING ACCONT TODAY!

Use promo code: CFDJ04H and receive a FREE T-SHIRT!

BlueDragonCertified V.I.P. Hosting Partner

macromedia ALLIANCE PARTNER



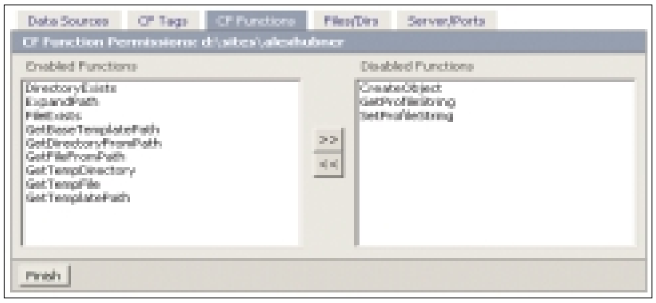
Data Sources | CF Tags | CF Functions | Files/Dirs | Server/Ports

CF Tag Permissions: d:\sites\alexhubner

Enabled Tags | Disabled Tags

Finish

Figure 6: Configuring the CF Tags tab



Data Sources | CF Tags | CF Functions | Files/Dirs | Server/Ports

CF Function Permissions: d:\sites\alexhubner

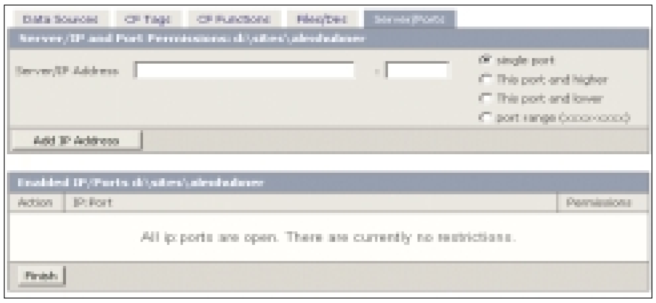
Enabled Functions:
DirectoryExists
ExpandPath
FileExists
GetBaseTemplatePath
GetDirectoryFromPath
GetFileFromPath
GetTempDirectory
GetTempFile
GetTemplatePath

Disabled Functions:
CreateObject
GetProfileString
SetProfileString

>> <<

Finish

Figure 7: Configuring the CF Functions tab



Data Sources | CF Tags | CF Functions | Files/Dirs | Server/Ports

Server/IP and Port Permissions: d:\sites\alexhubner

Server/IP Address: :

(●) single port
( ) This port and higher
( ) This port and lower
( ) port range (xxxx-xxxx)

Add IP Address

Enabled IP/Ports d:\sites\alexhubner

Action	IP:Port	Permissions
	All ip:ports are open. There are currently no restrictions.	

Finish

Figure 8: Configuring the Server/Ports tab

(IMHO) to a dedicated hosting environment rather than a shared one. It's easy to deduce that ColdFusion Server running in the "System" mode will be able to access and change registry entries, which is obviously not a good thing, so I recommend disabling this as well. Just remember that upon disabling this tag you'll need to change the storage type of your client variables in the "Client Variables" option of ColdFusion Administrator interface. "Cookies" are a good choice for shared environments.

- ***CFSchedule:*** Task scheduling is one of the great ColdFusion functionalities, but it should be used sparingly by your customers. Once it is poorly used, it can cause performance problems for the server. In shared environments where this tag is enabled it's very common to find scheduled tasks running at every minute and sometimes multiple times (if a programmer added it two or more times). The facility of scheduling tasks programmatically should be disabled and added only through a formal request by the customer. It's nice to observe a minimal interval between executions so as not to cause performance problems or scheduling "abuses" with heavy and bad scripts. The "CF Tags" tab should look as shown in Figure 6.

As for the CF Functions tab, I recommend disabling the following functions:

- ***CreateObject:*** More on this in the next section.
- ***GetProfileString and SetProfileString:*** These two functions are rarely used in typical CFML applications. They are generally used to change and read application's ".ini" files, including the ColdFusion server itself. For that reason I recommend disabling these functions.

The "CF Functions" tab should look as shown in Figure 7.

Concerns Regarding CFOBJECT, CFOBJECTCACHE, and CreateObject

I have suggested that CFOBJECT, CFOBJECTCACHE tags, and the function CreateObject should be disabled in a shared environment. In prior versions of ColdFusion, disabling such tags would not be a big problem. But in ColdFusion MX these tags play an important role because they are used by applications that consume ColdFusion Components (CFCs), Web services, and even Java classes in an increasingly popular way to write CFML as a pseudo-OO language. Be aware that by disabling these items you'll certainly be getting into trouble with some of your clients.

So why do I recommend disabling them? For a good reason: these tags involve security and privacy issues, which makes them good candidates for disabling in a shared environment. To explain why, it is necessary to briefly explain their functionalities:

As a pure Java application, ColdFusion offers an enormous gamut of possibilities, best of all being total integration with the Java runtime, including the execution of classes, EJBs, JSP custom tags, and many others. The interaction with the Java universe is done primarily through the CFOBJECT and CFOBJECTCACHE tags, along with the CreateObject function. These tags are also used to invoke COM, CORBA, and other components, but that is another story. This integration can be beneficial for many reasons, but it can also be problematic in shared environment.

ColdFusion stand-alone runs under the same JVM, therefore sharing Java resources with all users and applications on it. For that reason, it's possible for a user to invoke the Java API and classes that are not isolated within the sandbox (imagine bypassing CFFILE using java.io.file). It is also possible to access the ColdFusion Server ServiceFactory, which allows you to make changes on the server settings and read sensitive information about other applications running on the same server. Jochem van Dieten, a Team Macromedia member, offers helpful information about ColdFusion ServiceFactory at: http://spike.oli.tudelft.nl/jochemd/index.cfm?PageID=10. Using these tags and the CreateObject function, it is possible (as an example), to delete data sources or create new ones. You can also create ColdFusion mappings, schedules, and an infinity of other inherent resources of the ColdFusion Server (remember that ColdFusion is a Java application).

As I've mentioned before, the only way to securely support these tags in applications running in the same machine is through multiple ColdFusion instances (with multiple JVM instances). An important observation: clients may argue that without these tags enabled, they will not be able to use ColdFusion components. This is not true. Remember that CFCs can be invoked using the CFINVOKE tag, which is safe to enable.

Macromedia MAX 2004

The Power of Experience

Waves are the powerful result of critical mass, various elements coming together at a specific place and time. The experience of people coming together to shape ideas and share information is equally powerful. That's MAX: a powerful experience.

Learn the best techniques from industry experts; get current on new products; and share ideas in a forum dedicated to delivering the next wave of great websites and applications.

Experience

Choose from over 80 different hands-on and workshop sessions, in seven tracks, to create a schedule to meet your specific needs.

Connect

Share ideas with leading developers and designers at networking receptions, Birds-of-a-Feather sessions, and a special event. Retain your competitive edge at general sessions where you'll hear directly from Macromedia® and other industry leaders on what's next.

It all happens November 1–4 in New Orleans.
Be a part of it. Register today.
www.macromedia.com/go/max

Silver Sponsors

DoubleClick macromedia PRESS VitalStream

MAX
The 2004 Macromedia® Conference

The Server/Ports Tab

The last tab is the Server/Ports tab. The settings for this aren't very important in terms of security because the consumption of external resources is not an issue when the client will certainly be able to access his or her account via FTP. The Server/Ports tab, without restriction, should look as shown in Figure 8.

Conclusion

Whatever technology is offered by a hosted environment, there are many performance and security details to watch and care for. The ColdFusion Server is no different. Although this article focused solely on the creation of sandboxes and the reasons for their use in a shared environment, there are many other performance and security topics that are worth becoming familiar with. Read the product's documentation, participate in the mailing lists, read specialized blogs, stay in tune with best practices, and know the product to its core.

And last but not least: it's important to remember (although I believe I don't need to remind you) to install all existing patches and hotfixes for ColdFusion and your OS. If you still use ColdFusion MX 6.0, which was the first version of the MX line, you should (dare I say "must"?) upgrade to the most recent version, MX 6.1, which offers numerous bug fixes, security patches, and performance improvements – at no cost. You can download this version at the Macromedia Web site; your CFMX 6.0 serial number will work for this version.

About the Author

Alex Hübner is the co-coordinator of CFUG-SP Brazil and the manager of Technology at Navita (www.navita.com.br), a software development company based in São Paulo, Brazil. Alex is a Macromedia Certified Advanced ColdFusion developer and the author of the first non-English Macromedia related blog (www.cfgigolo.com). He is also responsible for the Portuguese (Brazilian) version of Ben Forta's CFFAQ.

alex@hubner. org.br

Problems Securing Your Applications?

— continued from page 7

The same holds true for the CF authentication/authorization tags and functions. I blogged about this at www.horwith.com.

After writing the TechNote, Sarge made an entry in his blog (found at www.sargeway.com/sarge/) that describes how using cookie storage for authentication (via the CFAPPLICATION "loginstorage" attribute) doesn't work properly either because sometimes when a session times out, the authentication cookie remains. Even though the session is empty, getAuthUser() still returns the last logged-in user's info (in the same browser). To expand on this it's also worth noting that if you authenticate against a "pure JAAS module" (e.g., a JAAS module written in Java, not CF) you are seen as logged in by ColdFusion; if you do a CFLOGOUT, it does not log you out of the Java application – even with common context roots and/or EAR/WAR deployment.

One last note about security blogs in the community – just so that it's not all about implementation problems per se. I also blogged about a somewhat easy fix to the problem, whether you're using pure Java or the CFML implementation (CFLOGIN/CFLOGINUSER/CFLOGOUT). The J2EE world frequently preaches use of design patterns – not too often do you get to see a good real-world example of a design pattern fixing a specific problem (though Hal Helms' article in this month's issue does happen to show one). Security in CFML applications is often accompanied by a "user" (or other) CFC instance persisted in the session scope. But wait – like JAAS, there are problems doing this in a clustered environment. When a session variable is created on one server in a cluster and "carries over" to another server in the cluster, it's known as "session replication". This allows Server A in a cluster to set a variable in the session scope and if, on a subsequent HTTP request, that same user's request went to Server B, the session variable would still exist.

In order for this replication to work, all variables stored in the session must implement the Java Serializable Interface. Unfortunately, ColdFusion Component Instances are not serializable, which means that if you want your CFML application to work properly in a clustered environment that uses replication, you cannot persist CFCs in the session scope. Fortunately, the J2EE "facade" pattern can be used to work around this limitation.

The idea is to create a CFC – a facade – that directly accesses the session scope. The CFC should have get/set methods for all of the values in your session. You then go to any CFC that your application puts an instance of in the session scope, and replace any references to variables that it's storing in the "variables" scope or the "this" scope with calls to get/set methods in the facade CFC. The last thing you'd want to do is stop persisting the CFC instance(s) in the application. There are two drawbacks to this approach. The first is a performance hit because your application now creates the instance(s) on every request. The second is that, from a design point of view, the data is no longer protected within an object. For the same reason that you shouldn't use the "this" scope in CFCs, you also shouldn't be storing object properties in the session scope rather than the object itself. "Session" and "this" are both scopes that may be accessed from outside the component, thus making it impossible to guarantee the integrity of the data or to enforce business logic when these values change.

If you want a simple example showing the facade pattern used in this way to get around the CFC serialization issue, I've put one at www.horwith.com/downloads/facade_fix_sessioncfc.zip. Note that I wrote this pretty quickly; it's a simple example, but it illustrates the idea. The code should be self-explanatory.

So, that does it for the first installment of **Community Focus**. Next month we'll examine more recent news in the community and will also begin preparation for MAX! We'll also be experimenting with more ***CFDJ*** issue and article formats, including more multipart articles and some other things I'd like to see

Seapine Software™
changing the world of software development

Complete source code and asset management in Dreamweaver MX—now possible with Surround SCM.

Dreamweaver users know a beautiful Web-based product is only skin deep. Underneath, it's a tangle of hundreds or thousands of ever changing source files. Without a good development process and strong tools, bad things happen. Surround SCM can help.

Surround SCM lets you...

Track multiple versions of your source files and easily compare and merge source code changes.

Check out files for exclusive use or work in private workspaces when collaborating on a team.

Automatically notify team members of changes to source files—push changes through your organization.

View complete audit trails of which files changed, why, and by whom.

Associate source code changes with feature requests, defects or change requests (requires additional purchase of TestTrack Pro).

Remotely access your source code repository from Dreamweaver MX.

Surround SCM adds flexible source code and digital asset control, powerful version control, and secure remote file access to Dreamweaver MX. Whether you are a team of one or one hundred, Surround SCM makes it easier to manage your source files, letting you focus on creating beautiful Web-based products.

Features:

Complete source code and digital asset control with private workspaces, automatic merging, role-based security and more.

IDE integration with Dreamweaver MX, JBuilder, Visual Studio, and other leading Web development tools.

Fast and secure remote access to your source files—work from anywhere.

Advanced branching and email notifications put you in complete control of your process.

External application triggers let you integrate Surround SCM into your Web site and product development processes.

Support for comprehensive issue management with TestTrack Pro—link changes to change requests, bug reports, feature requests and more.

Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X.

Want to learn more?

Download Seapine's just-released white paper, **Change Management and Dreamweaver**, to discover tips, tricks and best practices for achieving full software configuration functionality. Visit www.seapine.com/whitepaper.php and enter code WM0604 to receive your copy today.

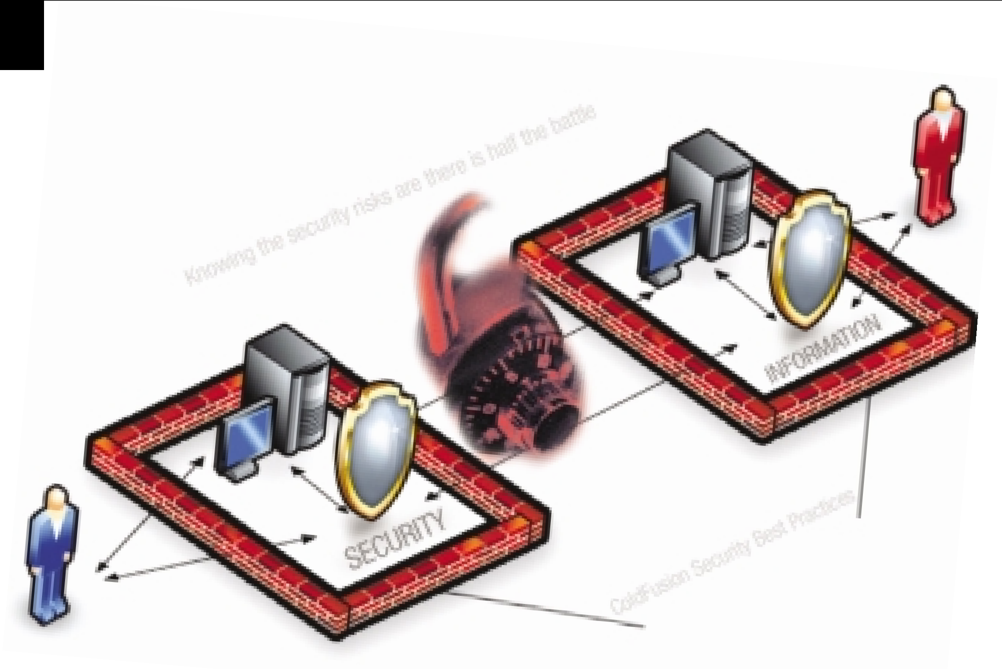
www.seapine.com
1-888-683-6456

Surround SCM Source Control

TestTrack Pro Defect Management

QA Wizard Automated Testing

macromedia ALLIANCE PARTNER


Knowing the security risks are there is half the battle
INFORMATION
SECURITY
ColdFusion Security Best Practices

ColdFusion Security Best Practices

Knowing the security risks are there is half the battle

By Bryan Murphy

The Internet has become a scary and hostile place; can your Web applications survive?

Although a lot of media attention has recently been paid to information security, surprisingly little has been published regarding ColdFusion security. Does this then mean that ColdFusion applications are immune to security risks? The answer, unfortunately, is no. Attacks may actually be easier to execute and much more prevalent than programmers would like to believe. Knowing the security risks are there is half the battle.

This article is not meant to be a silver bullet or a complete reference, as that could easily fill many volumes. I hope instead to give a thorough overview of ColdFusion security coding practices – thorough enough that you will know what types of things to take into consideration as you write your applications. Making your applications secure is probably a lot easier than you think.

I will cover some of the more common security problems, such as cross-site scripting, SQL injection, and man-in-the-middle attacks, along with some general ColdFusion security considerations. I will also touch on buffer overflow attacks and authentication and login security.

Cross-Site Scripting (XSS)

Perhaps the easiest type of attack to enact (and the easiest to prevent) against a Web application is that of a cross-site scripting (XSS) attack. Saving a form to the local machine the attacker is working from is the first step in this sort of attack. Once the file is there, the attacker then has the ability to change form field values that would not normally be accessible to a user simply filling out a form, such as radio buttons, select boxes, and hidden form fields, to name a few. The first thing an attacker would have to do is to change the post action of the form to include the full URL of your post page.

Preventing this kind of attack is as easy as checking for a referrer – or even just making sure that the post action is coming from your site. One way to accomplish this is to include code such as the following at the top of your posting pages:

```
<!--- XSS Protection --->
<cfif NOT len(cgi.http_referer)
    OR NOT findnocase(cgi.http_host,cgi.http_referer)>
      <h3>Post action aborted!</h3>
      <br />
      <p>Post from foreign host detected</p>
      <cfabort>
</cfif>
```

This code checks first to be sure that there is a referrer with the "NOT len(cgi.http_referer)". The reasoning behind this is that if the first page they are going to is the posting page, then something is obviously wrong. Also, this is a very real indicator that yes, this is an XSS attack. There can, however, be some false positives with this code because of older browser usage, but this is the fault of the browser and not the code. So at this point we know there is a referrer but not whether it is coming from our server, which is why we include the second part of the cfif statement "NOT findnocase(cgi.http_host,cgi.http_referer)". This section of the statement checks to be sure that this is indeed a page on your site going to post on your site. With this we know that it is most likely a legitimate transaction and will allow the user to post the form.

Note: Be aware of the "www." in your domain when using this code. You may want to alter the code to accommodate all of the possible domains for your site. It is also important to note that there is not a 100% guarantee on the referrer, as this information is sent from the client's browser and is therefore "spoofable."

SQL Injection Attacks

An injection attack is the act of embedding partial SQL queries inside of input that is sent to a WHERE clause in one of your existing queries. A classic (and all too common) example of vulnerable code would look like this:

```
<!--- SQL INJECTION VULNERABLE CODE -‡
<cfquery datasource="#myDNS#" name="qryLogin">
SELECT * FROM Users WHERE username= '#form.username#' AND pass-
word='#hash(form.password)#'
</cfquery>
<cfif qryLogin.recordCount gt 0>
  <cfset session.authenticated = 1>
<cfelse>
  Invalid login!
</cfif>
```

To exploit this an attacker would simply have to enter " ' or 1=1--" into the username field. That would produce an end query that looks like this:

```
SELECT * FROM Users WHERE username='' or 1=1-- AND password=''
```

As you can see, 1=1 will always be true. The "--" in MS SQL Server will comment out the rest of the query. The resulting record count will be however many records are in the Users table, obviously greater then 0. The user would be logged on and happily browsing your application.

There are a number of things developers can do to reduce the risk of SQL injection attacks in their applications. One technique I like to use is to write a UDF (user-defined function) that will escape out all quotes and then use it as a wrapper for all values passed to the query. Following is a simple example of such a UDF:

```
/**
 * UDF that replaces strings commonly used in SQL Injections
 * and replaces them with Unicode equivalents.
 *
 * Written for September 2004 issue of CFDJ magazine
 * Version 1 by Bryan Murphy, bryan@guardianlogic.com
 * Written in cfscript
 * @param string     Text to parse. (Required)
 * @return Returns a string.
 * @author Bryan Murphy (bryan@guardianlogic.com)
 * @version 1, July 22 2004
 */
function sqlSafe(string) {
  var sqlList = "-- ,'";
  var replacementList =

"#chr(38)##chr(35)##chr(52)##chr(53)##chr(59)##chr(38)##chr(35)##chr(52
)##chr(53)##chr(59)# , #chr(38)##chr(35)##chr(51)##chr(57)##chr(59)#";

  return trim(replaceList( string , sqlList , replacementList ));
}
```

The simplest way to protect your applications from a SQL injection attack would be to store all of your SQL code in stored procedures. Because a stored procedure does not dynamically create a SQL query, it is not susceptible.

ColdFusion also has the CFQUERYPARAM tag. Using this tag properly will also ensure the safe execution of your SQL queries. Be sure to provide the CFSQLType value to ensure that the data being passed is of the expected type for your database.

Man-in-the-Middle Attack (Session Hijacking)

Using this type of attack, the attacker sometimes sniffs the packets of a legitimate user. The attacker will then alter the packets and send them back at the application on the user's behalf.

The best way to eliminate this risk is by "tying" multiple variables from different scopes together. Require a session, database, and cookie variable to be concatenated and match a session-specific variable. An encrypted or hashed UUID works best for this purpose.

Parameter Manipulation

Another very common type of attack used against a Web application is parameter manipulation. Arguably just about any type of attack could be grouped into this category, but for the purpose of this article I will use this term to refer to attacks by users providing unexpected input into fields they are given access to.

These attacks could range from someone entering ".5" for the quantity of a $100 product and being charged only $50 for it, to someone entering a letter into a field that is expecting a number to see if they can receive an error message.

This type of attack simply comes down to not properly checking user-supplied attributes. If you are expecting a whole number, use the ceiling() function to round entries up to the nearest whole number; if you are expecting any sort of numeric value, use the val() function, which automatically strips out nonnumeric characters.

Your post actions should do a host of error handling and input validation before anything is saved to a database or any access is granted. Your error handling should be done server side. Do not rely on JavaScript or other client-side validation, as these are all easily manipulated by the end user. Be aware that CFFORM fields generate JavaScript validation, making it useless against this sort of attack.

Buffer Overflows

Buffer overflows are by far the most prominently publicized vulnerabilities. Although these are extremely prevalent in normal software, they are not a direct concern for ColdFusion developers, as the ColdFusion Server does not allocate memory at runtime or have pointers. This does not mean that the ColdFusion Server itself is not susceptible to such attacks. The recommended course of action is to subscribe to the ColdFusion security mail list (www.macromedia.com/devnet/security/security_zone/notification_service.html) and patch appropriately as patches are released.

General ColdFusion-Specific Security Considerations

There are some general things that every ColdFusion programmer should take into consideration. CFFILE, CFFTP, and CFPOP present a unique set of insecurities. Each of these tags allows an end user to write a file to your server. It is important to filter MIME types to exactly what you are expecting. Better yet, allow files to be saved only outside of Web-accessible directories.

In the event of an error, never give away any details (DSNs, table names, directory paths, and so forth) in the error message. Attackers will intentionally throw things at your application in an attempt to generate error messages that will aid them in their attacks.

SSL (Secure Sockets Layer) provides a layer of encryption between the client and the server. If your application stores or transfers any data of a sensitive nature (social security numbers, credit card numbers, etc.) you will *need* to use SSL. If it's a public site, your users will demand it. Even novice Internet users know to look for the little padlock in the corner of their Web browser; if they don't see it, they take their business to another company.

Authentication/Login Mechanisms

The last security issue I will touch on covers authentication systems and login systems. These systems may seem simple to the average developer, but creating a secure login requires a substantial amount of code. I would recommend putting a lot of time and effort into your login/authentication mechanism and placing it in a CFC or custom tag. Then you can reuse it across all of your applications, and if a problem is found it requires that you change code in only one place for all applications. It's a good idea to use and reuse trusted components. However, if you would prefer an off-the-shelf product, you may try MetaGuard (http://guardianlogic.com/?dna=products&rna=metaguard). We have written all of the laborious stuff for you.

Always hash/encrypt passwords and other sensitive data. If an attacker is somehow able to get direct access to your database, you don't want them to see all of your passwords in plain text.

Require complex passwords. Time and time again, end users have shown that they are completely content using the name of their pet, child, or even their username as a password. This is gold to an attacker, as it requires only a brief amount of research or one or two guesses to break it. I recommend using some sort of UDF to verify the complexity when a user goes to change or create a password. My favorite is passwordCheck(), which can be downloaded from www.cflib.org/udf.cfm?id=1072

Make passwords expire. This rule is sometimes disregarded, but it is still very important. If an attacker managed to gain a copy of your users table six months ago, he or she will still be able to gain access with those passwords today if you don't require expiration.

Log everything (or at least all that is useful). If someone is hammering an account with 100 invalid login attempts per minute and all you know is that your site is running slow, then your application has a way to go before it can be considered secure. Use CFLOG, write the events to a table in a database, or use CFFILE to write a plain text log file. Any or even combinations of these methods will work. Remember, these logs are useful only if someone actually checks them.

Set a maximum invalid login threshold lockout. In other words, if a user has X invalid login attempts, he or she will be locked out for Y minutes. Some applications require that an administrator unlock an account after a lockout occurs. In my opinion this is not the best method, as it could be used to create a DOS (denial of service) event. It is common for attackers to gather all of the logins they can for a site and run a brute-force or dictionary attack against each one. This will essentially lock out every user on your system (that they have usernames for). This is a nightmare to deal with.

Allow only one connection per user at a time. If someone attempts to log on as a user who is already logged on from a different IP, it should be logged as a security event. The user is either sharing passwords, or his or her password has been compromised.

Write IP-based auto-blacklisting for repeat offenders. This may even be the route you want to go for the invalid login lockout. This way the account itself remains usable by legitimate users, but the attacker's IP is locked out. Of course this does allow the attacker to make an attempt from another IP or to spoof the IP and try again.

As you can see, many security concerns must be taken into account when writing a Web application. It may seem like a lot of extra effort – until you experience your first XSS attack and the attacker makes off with your product for pennies or uses a SQL injection attack to steal your customer database, credit card numbers included.

Security considerations should be included in everything you do. They should be embedded into your thinking and not added as an afterthought.

Resources

- *OWASP:* www.owasp.org
- *Search Security:* http://searchsecurity.techtarget.com
- *Macromedia CF security:* www.macromedia.com/devnet/mx/coldfusion/security.html
- *CF server security announcement list:* www.macromedia.com/devnet/security/security_zone/notification_service.html
- *WASC:* http://webappsec.org

About the Author

Bryan Murphy is the owner of GuardianLogic, Inc. (www.guardianlogic.com), an information security firm that provides application and network vulnerability assessments and hardening. He is also one of the authors of Metazoa (www.metazoa.ca), a security-enhanced content management system; Membrane, an application-level firewall; and MetaGuard, a CFC that provides role-based login, authentication, and access control. Bryan has been an ethical hacker since the old-school BBS days. Visit his blog at www.downgrade.org.

bryan@guardianlogic.com

$99
per person

CF_Underground VI

October 31st 2004
10:00am - 5:00pm
New Orleans, LA
Check website for exact location and details

www.cfconf.org/cf_underground6/

Fill your brain with lots of cool CF programming meta tips & tricks, then relax and have a drink on us!

CF_Underground is the coolest pre-MAX event you'll attend during your visit.

Pick the brains of several experts like Simon Horwith, Sandra Clark, Shlomy Gantz, Hal Helms, Michael Smith, and more!

TeraTech Event
www.teratech.com
301.424.3903

Public-Key Encryption

Making strong encryption nearly painless

How secure are your applications? Public-Key encryption may be the solution when security really matters.

By Wayne Graham

If you have developed an application that requires user authentication, you have undoubtedly wrestled with varying levels of security. At a basic level, most security models revolve around membership, authentication, and authorization functions. Secure socket layers (SSL) is a popular method for securing the transmission of data between Web server and client. ColdFusion MX and ColdFusion 6.1 have very good integration with Java's Secure Socket Extensions Library, which is capable of 2048-bit encryption. While the transmission of the data over the Internet via SSL helps secure against electronic eavesdropping, the data stored in your applications may still be at risk.

The storage of passwords is a prime example of this security risk. If your database is compromised in some way, the attacker has access to all user accounts and passwords. As a result, programmers have developed various techniques for addressing this issue. ColdFusion itself has an encryption function available – encrypt() – that utilizes an XOR (exclusive OR) algorithm to generate a pseudo 32-bit symmetric key. Another method involves using ColdFusion's hash() function. The hash() function is based on an MD5 (message digest version 5) 128-bit hash algorithm that converts strings into 32-bit hexadecimal "fingerprint" or "message digest" representations of the original string. A stronger variant of this method involves introducing salt – a random string of some length – and concatenating it with the password before performing the hash function.

While storing an encrypted or hashed version of passwords using ColdFusion's built-in functions is a good practice, these methodologies fall a bit short when security is a real issue. The hash() function is a one-way encryption algorithm that can be decrypted only by brute force. MD5 hashing as a method of securing passwords and other data falls apart when one does a Google search of "MD5 crack." For unsalted hashes, the time needed to crack a single MD5 hash online is about 40 minutes (http://passcracking.com). Depending on your personal computer speeds, this can be done faster with a tool like md5crack (www.checksum.org/download/MD5Crack). In fact, in 1994 Paul van Oorschot and Mike Wiener showed that a brute force attack on a 128-bit hash function requires 2^{64} (2.10^{19}) evaluations to crack; at the time such a crack would take less than a month with a $10 million investment in hardware.

To deal with the shortcomings of 128-bit hash functions, stronger encryption algorithms have been invented. Today's 160-bit encryption algorithms such as SHA1 (secure hash algorithm, www.w3.org/PICS/DSig/SHA1_1_0.html) and RipeMD160 (www.esat.kuleuven.ac.be/~bosselae/ripemd160.html) increase the time required for a brute force attack. For areas where a 160-bit hash is still not strong enough, SHA also comes in 256-bit, 384-bit, and 512-bit data lengths for added security in one-way encryption.

Because hash() is a one-way encryption algorithm, it is most appropriate when text does not need to be read (as in the case of passwords). By contrast, the encrypt() function utilizes symmetric-key cryptography, meaning that both the sender and receiver of the string share a common key used to encrypt and decrypt the string. Thus, the private key must at some point be transferred in some secure way, and is only effective if the symmetric key is kept secret.

In ColdFusion, this transfer is done on the server in memory when a page with the encrypt() function is requested, which keeps the transmission of the passphrase reasonably secure. Yet, in the case of encrypt(),the key is actually passed in both the encrypt() and decrypt() functions as plain text:

```
<cfscript>
    password = "Th1s !s A R@alLy str0nG pA5Sw0rD!";
    symmetricKey = "pa$sPhrAs3 f0r 3ncRypt1ng p4s$w0rDs";

    encrypted = encrypt(password, symmetricKey);
    decrypted = decrypt(encrypted, symmetricKey);
</cfscript>

<cfoutput><p>#encrypted# <br/> #decrypted#</p></cfoutput>
```

Depending on who has access to your code, this could be a recipe for disaster.

When you need to be able to encrypt and decrypt, additional steps must be taken. ColdFusion's encrypt() function can be decrypted, but the key must be passed in the code on the server, causing a security issue (plus encrypted data placed on the Web can be fairly easily cracked using any number of free tools available on the Internet.

An alternative to ColdFusion's private-key encryption method is public-key encryption. Public-key encryption – or asymmetric encryption – requires two keys – one private and one public. Data encrypted with your public key can be decrypted only with your private key, allowing you to freely distribute your public key in a non-secure manner (i.e., as clear text posted on a Web page). Asymmetric encryption uses longer algorithms for calculating file fingerprints than symmetric encryption algorithms, and is effective for generating significantly obfuscated data. As a brief side note, these algorithms are processor intensive, so using public key encryption may not be appropriate for very large files.

Unfortunately, in order to take advantage of asymmetric encryption in ColdFusion, you must look beyond built-in ColdFusion tools. The two big players in the realm of public-key cryptography are Pretty Good Privacy (PGP; www.pgp.com) and GNU Privacy Guard (GnuPG; http://www.gnupg.org). "GnuPG is a complete and free replacement for PGP," and since GnuPG does not depend on the patented International Data Encryption Algorithm (IDEA), there are no restrictions on its use, nor are there any licensing fees for integrating GnuPG into your applications. This last fact makes it an attractive candidate for developers, and is used in the examples for this article. Along with the strong two-way encryption algorithms (1024-bit DSA and ElGamal), GnuPG also supports stronger hashing functions (SHA1, RIPEMD160, and SHA256) for your one-way encryption needs.

Installation

The first step in integrating GnuPG for ColdFusion is downloading the GnuPG binaries (or source code, for the more adventurous). Binaries are available at www.gnupg.org/(en)/download/index.html for most operating systems (examples in this article are based on a Windows implementation). Once you have downloaded the binaries, it is a good idea to verify the signature or MD5 checksum on the file to validate the integrity of the download. If you already have a version of GnuPG (or PGP), import the public key from www.gnupg.org/(en)/signature_key.html and the appropriate signature file from the main GnuPG download page. Verify the signature file by opening the command prompt, navigating to the location of the compressed binaries and signature file, and typing:

```
gpg --verify "gnupg-w32cli-1.2.5.zip.sig"
```

You should see:

```
gpg --verify "gnupg-w32cli-1.2.5.zip.sig"
gpg: Signature made 07/26/04 05:48:55  using DSA key ID
57548DCD
gpg: Good signature from "Werner Koch (gnupg sig)
<dd9jn@gnu.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the
owner
Primary key fingerprint: 6BD9 050F D8FC 941B 4341  2DCC 68B7
AB89 5754 8DCD
```

You have a good signature from Werner Koch for this file. The warning stems from an additional step needed to trust the signature. To do this, you need to execute the "--edit-key" command:

```
gpg --edit-key "dd9jn@gnu.org"
```

You will see:

```
gpg (GnuPG) 1.2.5; Copyright © 2004 Free Software Foundation,
Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

Pub 1024/57548DCD          created: 1998-07-07 expires: 2004-
12-31 trust: -/-
(1). Werner Koch (gnupg sig) <dd9jn@gnu.org>

Command>
```

You will notice that there is a prompt named Command. At this prompt, type "trust" to bring up the trust options:

```
pub       1024D/57548DCD created: 1998-07-07 expires: 2004-
12-31 trust: -/-
(1). Werner Koch (gnupg sig) <dd9jn@gnu.org>

Please decide how far you trust this user to correctly
Verify other users' keys (by looking at passports,
Checking fingerprints from difference sources...)?

 1 = Don't know
 2 = I do NOT trust
 3 = I trust marginally
 4 = I trust fully
 5 = I trust ultimately
 m = back to the main menu

Your decision?
```

I chose "5" to ultimately trust the key, and verified my choice.

```
pub     1024D/57548DCD created: 1998-07-07 expires: 2004-
12-31 trust: f/-
(1). Werner Koch (gnupg sig) <dd9jn@gnu.org>
Please note that the shown key validity is not necessarily correct
Unless you restart the program.

Command>
```

Type "quit" to exit GnuPG's command prompt. Now, if you run the same verification as before (gpg --verify "gnupg-w32cli-1.2.5.zip.sig"), you will see:

```
gpg: Signature made 07/26/04 08:48:55 using DSA key ID
57548DCD
gpg: Good signature from "Werner Koch (gnupg sig)
<dd9jn@gnu.org>"
gpg: check the trustdb
gpg: checking at depth 0 signed=0 of(-/q/n/m/f/u)=0/0/0/0/06
gpg: next trustdb check due at 2004-12-31
```

Please note that this is a detached signature, so both the signature file and downloaded file need to be in the same directory.

If you do not have a version of GnuPG (or PGP), you will need an MD5 checker like FastSum (www.fastsum.com) to verify the file's fingerprint. The command for checking the MD5 checksum will look like this:

```
C:\fastSum\fastSum.exe locationOfGnuPGDownload\gnupg-
w32cli-1.2.5.zip
```

The result will look like this:

```
MD5 Checksum calculation and verification utility. [1.6.0.92] EN
(C) 2003 Kirill Zinov and Vitaly Rogotsevich. Web site: www.fast-
sum.com

Calculating...

c:\downloads\gnupg-w32cli-1.2.5.zip
3D93D73942117C4C0182CB15E01DE70F

Calculation summary:
  Processed 1 files in 0 folders with total size 1.43 Mb.
  Elapsed time: 00:00:00 Average speed: 27.72 Mb\Sec.
```

Next, you need to verify the calculated checksum (3D93D73942117C4C0182CB15E01DE70F) against the MD5 Sum Summary that GnuPG provides at www.gnupg.org/(en)/download/integrity_check.html. For the Windows GnuPG 1.2.5 ZIP compressed file, the correct MD5 sum is 3d93d73942117c4c0182cb15e01de70f (case is not important). Because the strings match, you have verified the integrity of the downloaded file and can proceed to installing GnuPG.

You need to extract the GnuPG binaries to a secure place on your hard drive (i.e., somewhere that is not directly accessible to the Internet). The default location for Windows installations is c:\gnupg. *Note:* If you choose a location other than c:\gnupg you must edit the registry file gnupg-w32.reg (included in the binary distribution) to reflect the new location of the binaries; don't forget to execute the registry edits! An additional, and very useful, step is to add the path to the GnuPG binaries to your system PATH variable. Though not required, this step saves time when dealing with GnuPG at the command line, as you simply type "gpg" from any directory to launch GnuPG.

Once you have finished the initial verification, unpacking, and system configuration, you can verify your installation by opening the command prompt, navigating to the directory where you extracted the GnuPG binaries, and typing:

```
gpg --version
```

If everything was done correctly, you should see:

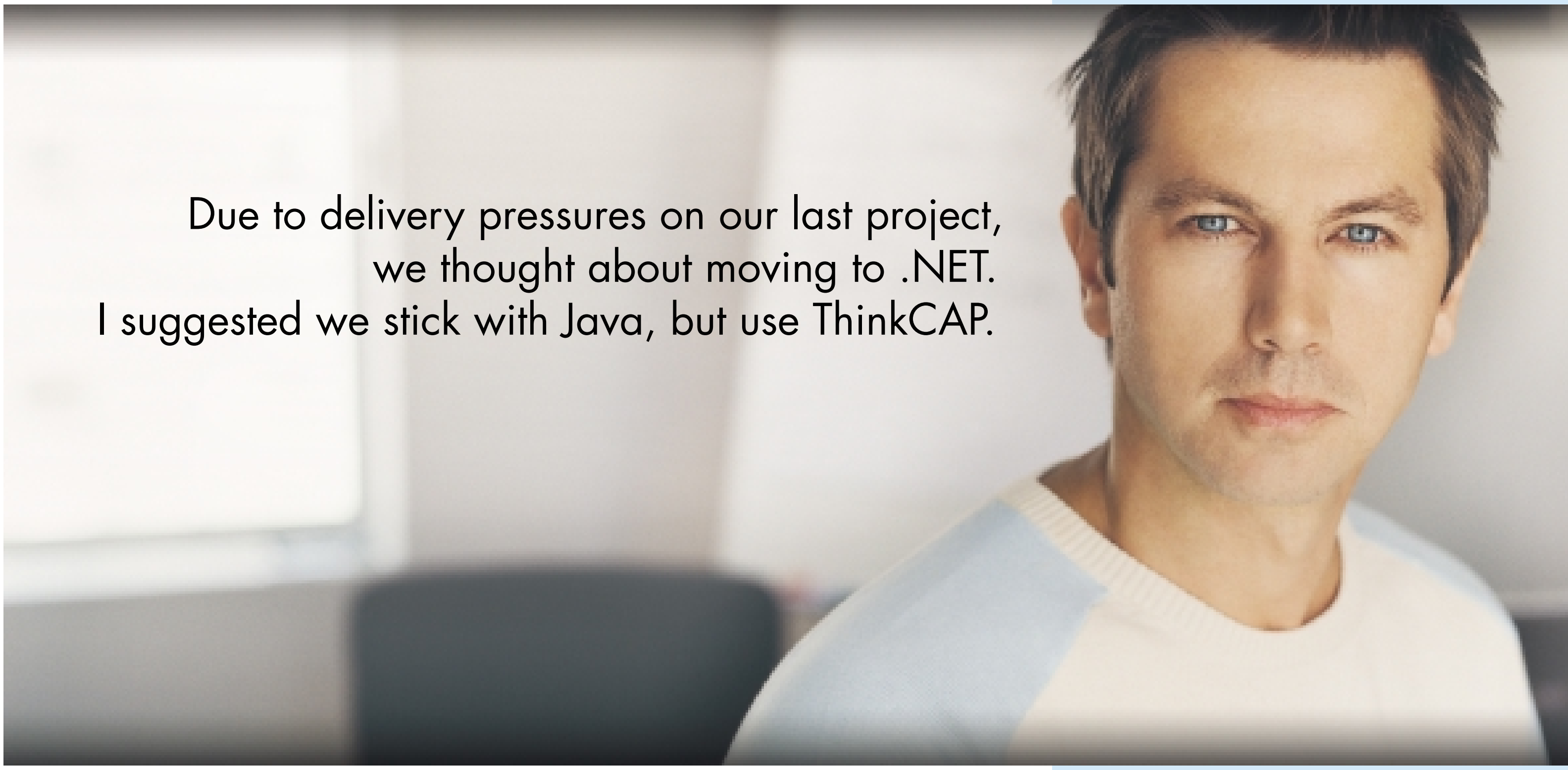
```
gpg (GnuPG) 1.2.5
Copyright (C) 2004 Free Software Foundation, Inc.
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions. See the file COPYING for details.

Home: c:/gnupg
Supported algorithms:
Pubkey: RSA, RSA-E, RSA-S, ELG-E, DSA, ELG
Cipher: 3DES, CAST5, BLOWFISH, AES, AES192, AES256,
TWOFISH
Hash: MD5, SHA1, RIPEMD160, SHA256
Compression: Uncompressed, ZIP, ZLIB
```

The next thing to do is to create your key pair. Because both your public and private keys are stored on your computer, GnuPG encrypts your private key further with a passphrase. When you use your private key, GnuPG prompts you for your passphrase, decrypts your private key in memory, and then uses it. To protect against dictionary attacks it is important to choose a "good" pass phrase for your private key (i.e., something that is not easily guessed by a human or brute force attack).

To ensure that you have a properly randomized passphrase, you may want to consider Diceware (http://world.std.com/~reinhold/diceware.html). Diceware theory uses dice to generate random words from a list of about 8,000 words. First, download the Diceware wordlist from http://world.std.com/~reinhold/diceware.wordlist.asc; an alternative list that removes some of the Americanisms from the list is at http://world.std.com/~reinhold/beale.wordlist.asc. After you have a list (or both), find a die and roll it five times, writing the results rolled to create a random five-digit number. Locate the corresponding word from the word list and write that word down. Diceware recommends repeating this process five times (25 rolls) to produce an optimal

Think .NET development is more productive than J2EE?

Think **again**.


Due to delivery pressures on our last project,
we thought about moving to .NET.
I suggested we stick with Java, but use ThinkCAP.

Think **better**.

ThinkCAP™

ClearNova's ThinkCAP is a comprehensive application platform that simplifies and accelerates the development and maintenance of J2EE-based business applications by 50 to 70%.

ThinkCAP's visual & intuitive designers bring high productivity to business developers (those with VB or PowerBuilder-like skills), content owners, and administrators while allowing J2EE architects & programmers to leverage its component infrastructure and build business logic using the tools and approaches they prefer. ThinkCAP utilizes existing infrastructure, web services, legacy systems, and business applications.

ThinkCAP saves organizations time and money—and lowers project risks. Applications are written faster and require less maintenance. Project teams utilize in-house skills and require less training. Existing infrastructure and application servers are leveraged.

With ThinkCAP you can build quality applications faster.

Learn more about ThinkCAP at www.clearnova.com/thinkcap

CLEARNOVA
APPLICATION DEVELOPMENT • SIMPLIFIED • ACCELERATED

Highly Visual Development Environment

MVC Framework with Page Flow & Actions

Advanced Data Aware Controls:
Forms, DataViews, Queries, Navigations
Workflows, Graphs, Treeviews, Grids, Tab

Smart Data Binding™ to data, objects, XML
sessions, or requests

Browser & server-side validation

Visual unit testing with RapidTest™

Service Flow Designer aggregates Web Services, EJBs, XML, and POJOs

Content Management engine & tools

Supports .NET clients

Integrated, seamless security

Use any app server or 3rd party tool

passphrase for most users (breakable only by organizations with large computing budgets). For instance, my rolls were as follows: 11234, 61262, 23212, 63352, and 42463

Looking these numbers up in the word list, I got “acorn timex ditto wade modem” as the passphrase for my private key.

Included in the CFC is a method to generate Diceware passphrases. The method implements Java’s java.security.SecureRandom package to provide cryptographically strong pseudo-random numbers to query a copy of a Diceware word list and return a passphrase of the length you specify in the method call. As a brief side note, computers are incapable of producing truly random numbers; the java.security.SecureRandom does a minimal job of complying with the National Institute of Standards and Technology's “Security Requirements for Cryptographic Modules” (http://csrc.nist.gov/cryptval/140-2.htm), but to get a truly random passphrase, a die (or dice) must be used in conjunction with the Diceware word list.

You may want to keep the paper your passphrase is written on in a safe place. However, this also becomes a security issue in that people other than you could gain access to your key. It should be noted that currently in the United States, written documents (including passphrases and keys) can be subpoenaed. Further, the Patriot Act allows the use of the FBI’s Carnivore system, designed to perform electronic eavesdropping (with judicial oversight), and is rumored to have the capability to harvest passphrases—so use your passphrase only for good!

After generating your passphrase, the next step is to generate your private and public key rings. Navigate to where your GnuPG binaries are located and run the following command:

```
gpg –gen-key
```

An interactive menu appears with options for the type of key pair you want to create. Type “1” to accept the default of “DSA and ElGamal” (the other options, “DSA” and “RSA,” are for signing files only). You are next prompted for your desired encryption level for your “ELG-E” key pair; the default is 1024 bits, more than enough for most users. To accept this default, hit the enter key. Next, specify when the key will expire. For our purposes here, the “key does not expire” option will suffice, so hit the enter key again. You are asked if you are sure you want to do this. If you are sure, type “y” and hit the enter key.

Type your name as you would like it to appear and hit the enter key. Next, type your e-mail address (and enter). The last field is an optional comment field, which allows you to add a comment to the key entry to further identify you (bet there are a lot of John Smiths out there) without necessarily knowing an e-mail address. You are then prompted to review your entry and given the opportunity to change any of the fields. If everything is okay, type the letter “O” and hit the enter key.

Finally, your Diceware passphrase needs to be typed and verified. After all this data has been entered, GnuPG generates two database files (pubring.gpg and secring.gpg) to hold your keys. GnuPG will display the keys you have just generated, along with the generated key’s fingerprint:

```
public and secret key created and signed.
key marked as ultimately trusted.
pub     1024D/D0D63B4A 2004-08-09 cfTest (ColdFusion test
account) <test@nowhere.com>
   Key fingerprint = 7894 04AD B584 2DBD AE71  3101 6FEE
   8A8F D0D6 3B4A
sub     1024g/E614ABE6 2004-08-09
```

You do not need to write any of this information down; as long as you know your passphrase you will be able to use this key for encryption/decryption and hashing functions. With all of this done, it’s time to start hooking GnuPG up to ColdFusion!

Java Wrapper

There are several options for getting GnuPG to interact with ColdFusion. A common approach to extending ColdFusion with external objects is to write COM, CORBA, or Java wrappers; in the case of GnuPG, <cfexecute> is also an option. (However, <cfexecute> may not work as desired when returning large amounts of data results to the browser using the variable attribute.) My background is in Java, so I implemented the wrapper for GnuPG in Java.

The Java implementation is relatively straightforward, consisting of two objects: GnuPG and ProcessStreamReader. The ProcessStreamReader object’s purpose is to read the standard output generated by the GnuPG binaries. GnuPG.java is also very straightforward, with most of the methods simply constructing the commands to pass to the gpg executable. The main workhorse of the object is the private method runGnuPG, used for actually processing input/output.

Download GnuPG.jar from www.sys-con/coldfusion/sourcec.cfm and place the file in your classpath folder (WEB-INF/lib) or create a classpath variable in the ColdFusion Administrator. You will need to restart your ColdFusion server before you can use the classes in the GnuPG package.

CFC

The logic for executing GnuPG commands resides in the Java wrapper, so the methods contained within the cfGnuPG CFC (see Listing 1) are extremely short. In fact, after creating the Java object, each of the methods in the CFC simply call their Java counterpart in the Java archive, as shown by the following example:

```
<cffunction name="gpgEncrypt" access="public" displayname="Encrypt"
hint="Encrypts data streams using GnuPG." output="No">
<cfargument name="str" required="Yes" type="string" hint="Data stream
to encrypt.">
   <cfargument name="keyID" required="Yes" type="string" hint="User
   key to encrypt to.">

   <cfscript>
         return gpg.encrypt(arguments.str, arguments.keyID);
   </cfscript>
</cffunction>
```

Usage

You’re now ready to take a test drive of the GnuPG system in ColdFusion! All that is required to begin using the component on the server is knowing the absolute path to the GnuPG binaries. On Windows systems you can initialize the component like this:

```
<cfscript>
        gpg = createObject("component", "cfGnuPG");
        gpg.init("c:\gnupg\gpg.exe");
</cfscript>
```

After the GnuPG wrapper has been initialized, you have access to all of the methods contained in the component object. Invoking each of the methods then becomes almost effortless. To invoke the listKeys() method to display all the keys on your public key ring, you would simply create a variable that accesses the object and its method within the <cfscript> block:

```
        keys = gpg.listKeys();
```

<cfoutput> can then be used to return the results of GnuPG to the browser.

Encrypting a message is just as easy. Simply invoke the gpgEncrypt() method with the text you want to encrypt and the key you would like to encrypt the message to.

```
   encrypted = gpg.gpgEncrypt("message", "recipient's key");
```

Decrypting a message is also simple (if you know the passphrase). Just as when using gpgEncrypt() to encrypt a message, gpgDecrypt() takes two parameters: the encrypted string and the passphrase. To decrypt the above example, you would simply code this:

```
   decrypted = gpg.gpgDecrypt(variables.encrypted, "passphrase");
```

To allow your users to provide their own public key to your public keyring, the importKey() method is particularly useful. By passing the importKey() method the public key you are able to store it for use on your public key ring. If the key is invalid, an error data stream is returned:

```
   badKey = gpg.importKey("not a key");
<cfoutput><pre>#badKey#</pre></cfoutput>
```

which generates the following message:

```
gpg: no valid OpenPGP data found.
gpg: Total number processed: 0
```

If you would like to maintain your user's keys on the server, you can generate key pairs by invoking the newKey() method. This method takes several arguments: a real name, comment (helps differentiate people of the same name), e-mail address, expiration date (see comments for usage), and a passphrase. Again, since this method passes sensitive data, as do most methods in the GnuPG object, it is important to secure the transmis-

HostMySite.com™

Built for ColdFusion Pros by ColdFusion Pros

- 24 / 7 / 365 Phone Support
- 99.9+% Uptime
- Macromedia Alliance Partner
- "Full Control" Panel
- CFMX 6.1 or CF 5.0
- SQL Server 2000 or 7.0
- Custom Tags Welcome

plans from $8.95 / mo.

FREE Domain Name*
FREE Setup
FREE 2 Months

Visit www.HostMySite.com/cfdj for:
2 Months Free
FREE Setup and FREE Domain Name
*on any annual shared hosting plan

call today 877•248•HOST (4678)

sion of that data via SSL. The newKey() method does take a few seconds to complete, as GnuPG implements complex mathematical algorithms to generate your public and secret keys.

Signing data is another important aspect of this wrapper. Signing allows others to verify that a data stream was not tampered with, be it a Web page, source file, image, or any other type of data stream. As with encrypt and decrypt, you utilize the sign/verify methods in the CFC.

```
signed = gpg.sign("Text to sign.", "passphrase");
verify = gpg.verify(signed);
```

The output from the verify variable will notify you if the signature matches the key ring. One potential use of this is to sign download files that you might host on your site. The examples included with this article show you how to create a file upload system that automatically signs files that you might have on your Web site for download. Also included, is a simple e-mail encryption example that allows you to quickly integrate encrypted e-mail into your existing ColdFusion e-mail application. Examples using this CFC and Java wrapper can be downloaded from www.sys-con.com/coldfusion/sourcec.cfm.

Conclusion

The CFC and Java wrappers for Gnu Privacy Guard make it nearly painless to integrate strong encryption into your programming projects. However, if you do not take the proper steps in encrypting transmission, or remain vigilant in your security implementation using strong key encryption methods as provided here, your work will be for naught. GnuPG does a very good job at securing data, as long as the passphrase is kept secret. Developing an encryption system that does not require user intervention will require the storage of passphrases in a database or other persistence mechanism. It is essential that this security issue be addressed during the design phase of your project, and that steps be taken to properly obfuscate the passphrases.

References

- van Oorschot, P., and Wiener, M. 1994. "Parallel Collision Search with Applications to Hash Functions and Discrete Logarithms." *2nd ACM Conference on Computer and Communications Security.* ACM Press.
- *The Diceware Passphrase FAQ:* http://world.std.com/~reinhold/dicewarefaq.html#howlong

About the Author

Wayne Graham is a systems administrator at the College of William and Mary's Earl Gregg Swem Library. Wayne is also the comanager of the Williamsburg Macromedia User's Group and has been developing with ASP, Java, and ColdFusion since 2001.

wsgrah@wm.edu

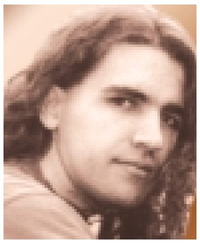
Listing 1

```
<cfcomponent displayname="cfGnuPG"
hint="ColdFusion wrapper for Java wrapper of
GnuPG.">

 <cfproperty name="gpgRuntime" type="string"
default="gpg" required="true" hint="Absolute
path of GnuPG runtime binaries.">

 <cffunction name="init" access="public" out-
put="No">
   <cfargument name="gpgRuntime" required="No"
type="string" default="gpg" hint="Absolute path
to GnuPG runtime binary. If none passed to the
constructor, the value ""gpg"" is used. Make
sure this is in your PATH variable if you are
going to use this.">
   <cfscript>
     gpg = createObject("java",
"com.sys_con.ColdFusion.GnuPG");
     gpg.init(arguments.gpgRuntime);
   </cfscript>

 </cffunction>

 <cffunction name="getRuntime" access="public"
displayname="Get GnuPG Runtime path."
output="No" returntype="string" hint="Returns
the GnuPG runtime path.">
   <cfscript>
     return gpg.getGnuPGRuntime();
   </cfscript>
 </cffunction>

 <cffunction name="newKey" access="public" dis-
playname="New Key" output="No"
returntype="string" hint="Creates a new GnuPG
key pair. <span class=""color:
##ff0000;"">Warning:</span> Generating key
pairs is slow...be patient.">
   <cfargument name="realName" required="Yes"
type="string" hint="Name for the person for
whom the key is being generated.">
   <cfargument name="comment" required="Yes"
type="string" hint="Name comment (to help dif-
ferentiate people with similar names).">
   <cfargument name="email" required="Yes"
type="string" hint="Email address for the
key.">
   <cfargument name="expireDate" required="Yes"
type="string" hint="Valid values:<ul><li>0 -->
key does not expire</li><li><em>n</em> --> key
expires in n days</li><li><em>n</em>w --> key
expires in n weeks</li><li><em>n</em>m --> key
expires in n months</li><li><em>n</em>y --> key
expires in n years</li></ul>">
   <cfargument name="passphrase" required="Yes"
type="string" hint="Private passphrase for the
key pair.">

   <cfscript>
     return gpg.newKey(arguments.realName,
arguments.comment, arguments.email,
arguments.expireDate, arguments.passphrase);
   </cfscript>
 </cffunction>

 <cffunction name="gpgEncrypt" access="public"
displayname="Encrypt" hint="Encrypts data
streams using GnuPG." output="No">
   <cfargument name="str" required="Yes"
type="string" hint="Data stream to encrypt.">
   <cfargument name="keyID" required="Yes"
type="string" hint="User key to encrypt to.">

   <cfscript>
     return gpg.encrypt(arguments.str, argu-
ments.keyID);
   </cfscript>

 </cffunction>

 <cffunction name="gpgDecrypt" access="public"
displayname="Decrypt" hint="Decrypts data
streams using GnuPG." output="No">
   <cfargument name="str" required="Yes"
type="string" hint="Data stream to decrypt.">
   <cfargument name="passphrase" required="Yes"
type="string" hint="Passphrase for decrypting
data stream.">

   <cfscript>
     return gpg.decrypt(arguments.str, argu-
ments.passphrase);
   </cfscript>

 </cffunction>

 <cffunction name="sign" access="public" dis-
playname="Sign" hint="Creates a signature fin-
gerprint." output="No">
   <cfargument name="str" required="Yes"
type="string" hint="Data stream to sign.">
   <cfargument name="passphrase" required="Yes"
type="string" hint="Passphrase for signature.">

   <cfscript>
     return gpg.sign(arguments.str,
arguments.passphrase);
   </cfscript>

 </cffunction>
 <cffunction name="signFile" access="public"
displayname="Sign File" hint="Creates a signa-
ture file for a given file.">
   <cfargument name="filePath" required="Yes"
type="string" hint="Path of file to sign.">
   <cfargument name="passphrase"
required="Yes" type="string" hint="Passphrase
for signature.">
   <!--- create a Java-safe path --->
   <cfset filePath = replace(filePath, "\",
"\\", "all")>

   <cfscript>
     return gpg.signFile(arguments.filePath,
arguments.passphrase);

   </cfscript>
 </cffunction>

 <!--- signature block not being returned --->
 <cffunction name="clearSign" access="public"
displayname="Clear Sign" hint="Creates a clear
signed fingerprint for a data stream.">
   <cfargument name="str" required="Yes"
type="string" hint="Data stream to sign.">
   <cfargument name="passphrase"
required="Yes" type="string" hint="Passphrase
for signature.">

   <cfscript>
     return gpg.clearSign(arguments.str, argu-
ments.passphrase);
   </cfscript>
 </cffunction>

 <cffunction name="signAndEncrypt"
access="public" displayname="Sign and Encrypt"
hint="Signs and encrypts a datastream." out-
put="No">
   <cfargument name="str" required="Yes"
type="string" hint="Data stream to sign.">
   <cfargument name="keyID" required="Yes"
type="string" hint="User key to encrypt to.">
   <cfargument name="passphrase"
required="Yes" type="string" hint="Passphrase
for signature.">

   <cfscript>
     return gpg.signAndEncrypt(arguments.str,
arguments.keyID, arguments.passphrase);
   </cfscript>
 </cffunction>

 <cffunction name="verifySignature"
access="public" displayname="Verify Signature"
hint="Verifies a given signature's validity.">
   <cfargument name="sig" required="Yes"
type="string" hint="Signature to verify.">

   <cfscript>
     return
gpg.verifySignature(arguments.sig);
   </cfscript>
 </cffunction>

 <cffunction name="verifyFile" access="public"
displayname="Verify File" hint="Verifies a
given file's signature validity.">
   <cfargument name="filePath" required="Yes"
type="string" hint="Path to file to verify.">

   <cfscript>
     return
gpg.verifyFile(arguments.filePath);
   </cfscript>
 </cffunction>

 <cffunction name="listKeys" access="public"
displayname="List Public Keys" hint="Returns
all keys in hte public key ring." output="No">
   <cfscript>
     return gpg.listKeys();
   </cfscript>
 </cffunction>

 <cffunction name="getKey" access="public"
displayname="Get Key" hint="Returns specified
key from the public key ring.">
   <cfargument name="keyID" required="Yes"
type="string" hint="KeyID to display">

   <cfscript>
     return gpg.listKeys(arguments.keyID);
   </cfscript>
 </cffunction>

 <cffunction name="listSecretKeys"
access="public" displayname="List Secret Keys"
hint="Returns all keys in the secret key
ring." output="No">
   <cfscript>
     return gpg.listSecretKeys();
   </cfscript>
 </cffunction>

 <cffunction name="getSecretKey" access="pub-
lic" displayname="Get secret key"
hint="Returns the specified key from the
secret keyring." output="No"
returntype="string">
   <cfargument name="keyID" required="Yes"
type="string" hint="KeyID to display">

   <cfscript>
     return
gpg.listSecretKeys(arguments.keyID);
   </cfscript>
 </cffunction>

 <cffunction name="finger" access="public"
displayname="List fingerprints" hint="Returns
all public keyring fingerprints." output="No"
returntype="string">
   <cfscript>
     return gpg.finger();
   </cfscript>
 </cffunction>

 <cffunction name="getFinger" access="public"
displayname="Get fingerprint" hint="Returns
the specified fingerprint from your public
keyring." output="No" returntype="string">
   <cfargument name="keyID" required="Yes"
type="string" hint="KeyID to display">

   <cfscript>
     return gpg.finger(arguments.keyID);
   </cfscript>
 </cffunction>

 <cffunction name="listSigs" access="public"
displayname="List Signatures" hint="Returns
all signatures from your public keyring.">
   <cfscript>
     return gpg.listSigs();
   </cfscript>
 </cffunction>

 <cffunction name="getPublicKey" access="pub-
lic" displayname="Get Public Key"
hint="Returns the public key for the specified
keyID.">
   <cfargument name="keyID" required="Yes"
type="string" hint="KeyID to display">
   <cfscript>
     return gpg.getPublicKey(arguments.keyID);
   </cfscript>
 </cffunction>

   <cffunction name="importKey" access="pub-
lic" displayname="Import public key."
hint="Imports a key into your public keyring."
returntype="string">
   <cfargument name="key" required="Yes"
type="string" hint="Key to import">

   <cfscript>
     return gpg.importKey(arguments.key);
   </cfscript>
 </cffunction>

 <cffunction name="dicewarePassphrase"
access="public" displayname="Diceware
Passphrase" hint="Generates a Diceware
passphrase using java.security.SecureRandom."
returntype="string" output="No">
   <cfargument name="length" default="5"
required="Yes" type="numeric">

   <cfset var number = "">
   <cfset var results = "">

   <!--- get the word list --->
   <cfhttp
url="http://support.swem.wm.edu/tools/wordlist
.txt" method="GET" name="wordlist" delimiter="
" textqualifier=" "></cfhttp>

   <!--- create words --->
   <cfloop from="1" to="#arguments.length#"
index="i">

     <cfset number = rollDie(5)>

     <!--- query wordlist for word --->
     <cfquery name="word" dbtype="query">
       SELECT word
       FROM wordList
       WHERE number = '#number#'
     </cfquery>

     <cfset results = results & " " &
word.word>
   </cfloop>

   <cfreturn results>

 </cffunction>

 <cffunction name="rollDie" access="private"
displayname="Roll Dice" hint="Similates
rolling dice to generate passphrases" out-
put="No">
   <cfargument name="rolls" required="No"
default="5" type="numeric" hint="Number of
rolls">

   <cfscript>
     //create a Random number
     rand = CreateObject("java", "java.securi-
ty.SecureRandom");
     result = "";

     //loop over to create a 5 digit number
     for(i = 0; i LT arguments.rolls; i = i +
1){
       result = result & abs(rand.nextInt())
MOD 6 + 1;
     }

     return result;
   </cfscript>
 </cffunction>

</cfcomponent>
```

Download the Code...
Go to www.coldfusionjournal.com

ColdFusion's Application Framework

Apply it to your development



By Jeffry Houser

For the past month or so, I've had the pleasure of doing some development in Lotus Domino. I worked with Domino a lot when I was just starting out my IT career, so I know it's a powerful development platform for client/server applications.

I'm sure if you're reading this, you're developing, or want to be developing, client/server applications with ColdFusion. In the Domino world, we have the Domino server and Lotus Notes client. When developing ColdFusion apps, your client is most likely Internet Explorer, Safari, or Navigator. Your Web server is probably IIS or Apache.

While juggling some Notes work with my standard Web fare, I started thinking about how the Lotus Notes client is different from a browser client. The Notes client is smart. It knows that I like my e-mail sorted by ascending date. It knows which documents I've read and which I haven't. Lotus Notes maintains my state in the application.

The browser, on the other hand, has no idea about any of these things. Every request exists in a vacuum and has no knowledge of any other request. This is a problem that plagues all Web applications, whether you're accessing your gmail account or changing your MSN home page preferences. How does the browser know what's going on? It doesn't, but thankfully in ColdFusion there is an easy way to handle this. This article will talk about ColdFusion's application framework.

cfapplication Tag

At the heart of ColdFusion's application framework is the cfapplication tag. This tag defines how ColdFusion stores application-specific and session-specific data. An application is a group of related actions or functions that work together to give the user a seamless experience. Any of the Web-based e-mail accounts, such as gmail, Yahoo Mail, or Hotmail would serve as a perfect example of a Web application. Even though there is only one application, there can be many users on it. A session keeps track of individual users.

These are the attributes of the cfapplication tag:

- ***Name:*** The name attribute is used to accept the name of the application. ColdFusion uses the name to keep application and session data separate within the context of an application. You can have as many applications as you like on a ColdFusion server, but each one must have a different name. It is not uncommon to have a site and the site administrator running as separate applications, even though they often access the same database.
- ***ApplicationTimeOut:*** The application timeout attribute specifies the life span of the application variables. You can specify the value of this attribute with the CreateTimeSpan function. More information on the CreateTimeSpan function can be found at http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/functi53.htm#wp1102715. A little-known fact is that any place that you use the CreateTimeSpan function can be replaced with a constant value. For example, one day is equal to 1. Two days and 12 hours equals 2.5. You can find the specific value you need by wrapping the CreateTimeSpan function in a cfoutput, executing the template, and using the value output. Using the CreateTimeSpan function is more explicit, but using an integer value is one less function that your application has to execute. The maximum value set in the ColdFusion administrator will take precedence if this value is larger than that value. The timeout value is specified from the date of last access.
- ***LoginStorage:*** The login storage attribute is used to specify where login information is stored. The login information is specified in ColdFusion's cflogin and cfloginuser tags. The default value is cookie, with the alternative being session.
- ***SetClientCookies:*** By default, ColdFusion uses two values to associate sessions on the server with a specific client. The first value is CFID, which is an integer that increments. The second is CFTOKEN, which is a random number. The setClientCookies attribute tells CF whether to store these values as cookies. If set to no, you'll have to pass the CFID and CFTOKEN in the query string on each request to access session information. These values must be in the query string, so on form posts you must add them to the action attribute of the form tag, not pass them as hidden form variables. To make this easier, there is a session variable called URLToken that contains the query string with CFID and CFTOKEN. If you have J2EE sessions enabled in the ColdFusion administrator, ColdFusion will set a JsessionID value instead of CFID and CFTOKEN.
- ***ClientManagement:*** The ClientManagement attribute is a Boolean value that tells the application whether or not to allow for client variables in the application. Client variables

are similar to session variables, except that they are stored on disk instead of in memory. Client variables are better in clustered environments, because the data is accessible from all servers when stored in cookies or a central database.

- ***ClientStorage:*** The ClientStorage attribute is used to specify the location where client variables are stored. There are three options here. The default is to store them in the registry. I wouldn't recommend ever using the registry. The registry is not designed for massive read and writes. I've also heard too many horror stories of the registry growing too big and bringing down machines. The second option is to store the client variables as browser cookies. This has advantages, such as the cookies will always exist when the user returns to the site. If you use cookies, there's a limit to the amount of data a single site can store as cookie data. The third option is my preferred (and the recommended) choice, which is to store the information in a database. You can specify the name of the data source for the value of this attribute. Storing client variables in a database is much slower than using browser cookies; however, your only size limitation is the amount of hard disk space you have. You can set the default client variable storage options in the ColdFusion administrator, although they can be overridden by the cfapplication tag.
- ***SessionManagement***: The SessionManagement attribute is a Boolean attribute that specifies whether session management is enabled for this application or not. The default is no.
- ***SessionTimeout:*** The SessionTimeout attribute is similar to the ApplicationTimeout attribute. It specifies how long the session variables will exist. Its value can be set with the CreateTimeSpan function. The maximum value set in the ColdFusion administrator will take precedence if the attribute value is larger than that CF Admin value. The timeout value is referenced from the date of the last request.
- ***SetDomainCookies:*** The SetDomainCookies attribute is used on clustered servers. Instead of setting the cookies for the host, it sets them for the full domain. That means when a user switches from one machine to another, ColdFusion will still know about the user's session. Keep in mind that session variables are not stored between requests; however, client variables stored as cookies or in a data source are, assuming that both machines have access to the same data source, of course.

In a typical cfapplication tag, you'd probably use:

```
<cfapplication name="MyApp"
applicationTimeOut="1" clientstorage="MyDSN"
setClientCookies="Yes" sessionManagement="Yes"
sessionTimeOut = ".4">
```

This will create the application MyApp with an application timeout of one day and a session timeout of one hour. ClientStorage will be in the MyDSN database, and the CFID and CFTOKEN values will be cookies on the user's machine. Domain cookies are not set, since that is the default behavior and we didn't specify the attribute. Login storage information will be stored as cookies, since we didn't specify that attribute either.

Application, Session, and Client Variable Scopes

One of the primary reasons for setting up an application with the cfapplication tag is so you can store application and session-specific data between page requests. If you do not set a cfapplication tag in a page request, you can still use variables in the application, client, or session scopes. However, there will be no way to reference the values on your next page request. You should always set up your application using the cfapplication tag.

Because variables in the application, client, and session scopes persist between requests, you often need to set them only on application or session initialization, respectively. Unfortunately, ColdFusion does not have a way to execute code when a session or application is initialized. There is a great Java workaround for this. You can read more about it in "Making the Most of J2EE Event Listeners," by Eric Brancaccio in the May 2004 edition of ***CFDJ***. You may also want to tell Macromedia you want this feature to be in the next version of ColdFusion. If so, go to www.macromedia.com/support/email/wishform/.

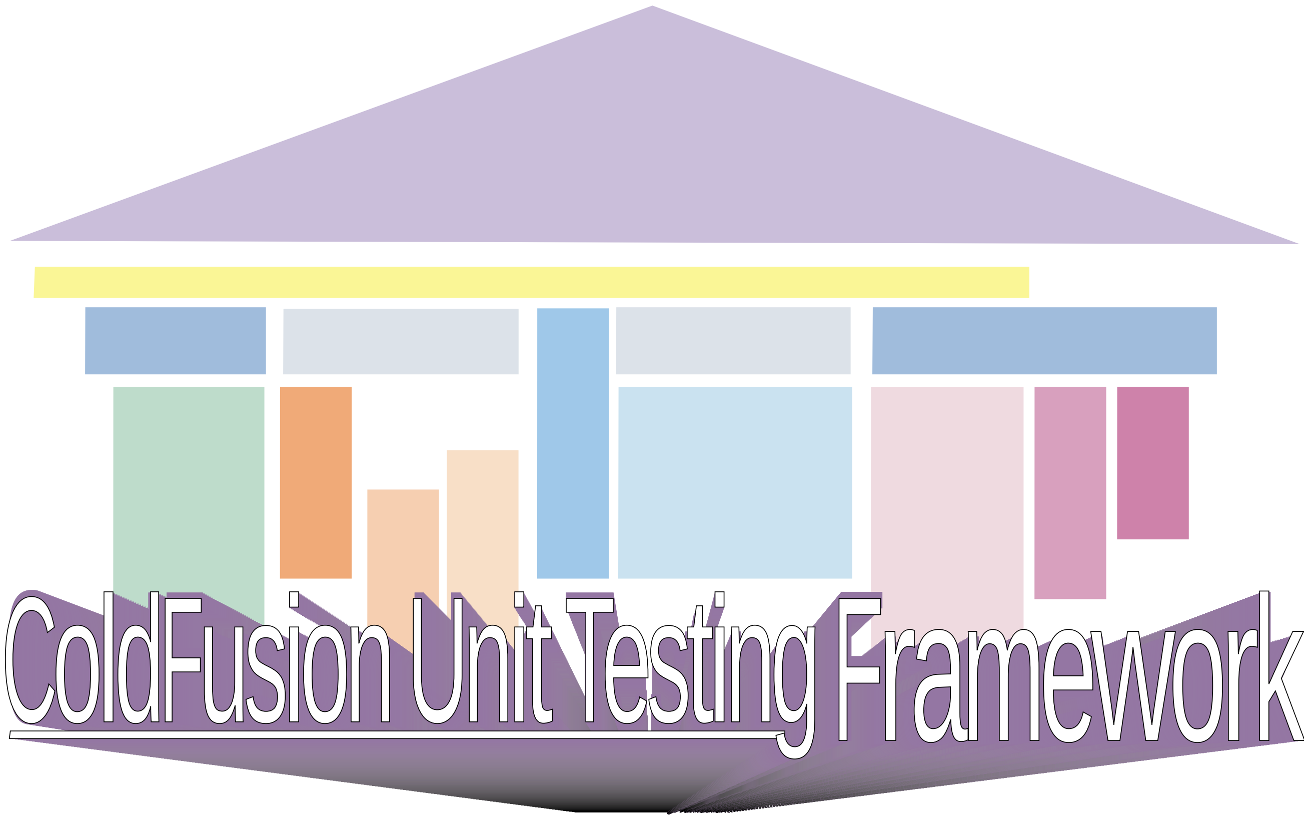
Until we have that feature, there is a simple method for simulating an application or session startup. You can use a cfif to see if one of your application, session, or client variables has been defined yet. This is a simple example:

```
<cfif not IsDefined("application.test")>
 <cfset application.test = "Not Set Yes">
</cfif>

<cfif not IsDefined("session.test")>
 <cfset session.test = "Not Set Yet">
</cfif>
```

The code utilizes the cfif tag (reference this column in the April 2004 issue), cfset tag (reference this column in February 2004), and the IsDefined function. The IsDefined function accepts one parameter, a string. The string is the name of the variable. If the variable is defined, the function returns true, otherwise it returns false. If the application variable is defined, you don't have to do anything. If it isn't, then you need to define it. The same concept applies to session variables. Generally, you only need to check for the existence of a single variable. If one is not defined, then it is safe to assume that none of them are defined.

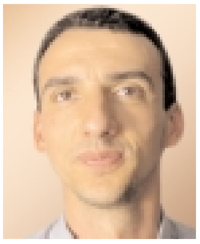
Session and application variables are called shared scope variables, because they are stored in shared memory space. You should always consider locking your shared scope variable access with the cflock tag. An in-depth explanation of locking is beyond the scope of this article. Read about the cflock tag in the live docs: http://livedocs.macromedia.com/coldfusion/6.1/htmldocs/tags-p71.htm#wp1100787. Versions of CF5 and prior had memory stability problems if you did not use locking. Read the Macromedia TechNote at www.macromedia.com/support/coldfusion/ts/documents/tn17882.htm. ColdFusion MX no longer has memory stability problems, but the best practices of locking remain relatively unchanged. You can find more information at www.macromedia.com/support/coldfusion/ts/documents/tn18235.htm.

–continued on page 35



ColdFusion Unit Testing Framework

Paul Kenney's version



By Harry Klein

Unit testing is receiving a lot of attention these days. Since Macromedia introduced CFCs (ColdFusion Components), developers have been intrigued by the new object-oriented possibilities ColdFusion offers. But OOP also has weaknesses – debugging applications got harder.

Many open source frameworks have already been borrowed from the "mother language," Java. If you examine the coldfusionmx/lib folder you will discover Java libraries like Crimson, jintegra, log4j, Xalan, and others.

cfcUnit is a unit testing framework that is derived from the JUnit framework available for Java. cfcUnit is a part of the OpenXCF project on SourceForge.net. This means that it is available as a true open source project under the Apache license. You can check it out at http://openxcf.sourceforge.net. The author, Paul Kenney, did a great job in porting this application to ColdFusion.

What Is Unit Testing?

The term "unit testing" was not very common in the ColdFusion development community before the release of CFMX. With the introduction of CFCs in ColdFusion MX, developers are looking for better ways to test their applications, and the JUnit example of a unit testing framework has a lot of appeal for CF developers. I will try to explain the goals of unit testing and why it is important for ColdFusion developers.

Unit tests are a key component of software engineering. They are programs written to run in test classes. Each typically sends a class a fixed message and verifies that it returns the predicted answer. Developers write tests for every class they produce. The tests are intended to test every aspect of the class that could conceivably not work. When developers go to release new code, they run all the unit tests. The tests must run at 100%. If any test fails, they figure out why and fix the problem. Of course, sometimes the tests let something slip through. When that happens, developers unconditionally enhance the unit tests so that problem, and any similar one that comes to mind, won't happen again.

Unit testing is especially useful for business logic (not presentation code), which can be tested programmatically rather than requiring user interaction with an interface. Unit testing is also important if you reorganize your code. If you've already written test scripts then it's easier and quicker to test business logic after changes are made (because the test scripts already exist).

Using a testing framework increases the chances that developers will actually write test code. A good testing framework should meet the following expectations:

- It should be easy to use.
- It should create tests that can be repeated.
- It should allow tests to be executed not only from the original author.
- It should support tests that can be combined and do not interfere with one another.

As I mentioned in the introduction, cfcUnit is a port from the well-known Java unit testing framework JUnit. It is written entirely in ColdFusion Components and based on the API of the JUnit testing framework.

Developers should also understand the limitations of unit testing. Unit testing adversaries argue that it's a waste of time because writing unit tests adds a lot of time to the development process.

In addition, many developers don't like the fact that unit tests never check the interconnections between components within an application.

Installation Steps

Before installing cfcUnit, you must be running CFMX 6.1 and the Mach-II (1.0.9) application framework. Prior versions of CFMX will not work and you should use the latest Mach-II version.

Mach-II is a framework for building robust, maintainable software. You can read more about Mach II on the official Mach II Web site (www.mach-ii.com/) and Sean Corfield's unofficial Mach II Web site (www.corfield.org/machii/).

If you are wondering why Mach-II is required, a Mach-II test runner application is included in the cfcUnit download file. A test runner is an application that provides a framework for performing unit tests. This could be a desktop application, an applet (JUnit contains a swing test runner), or a Web application.

Paul Kenney is working on a non-Mach-II test runner so that the idea of using Mach-II doesn't get in the way of understanding how cfcUnit works.

First you should download the latest version of cfcUnit (v0.8.3) from the cfcUnit Web site – www.cfcunit.org.

Then simply unpack the Zip file into your webroot. A new directory structure "cfcUnit" will be created. cfcUnit should now work for you.

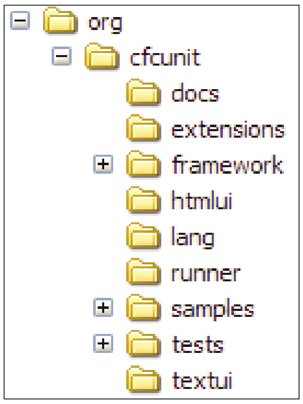


Figure 1: cfcUnit folders

fast . easy . affordable .

CommonSpot™ Content Server

Efficient Content Management

- 100 % browser-based
- Content object architecture
- Template driven pages
- 50+ standard elements
- Extensible via ColdFusion
- Content reuse
- Content scheduling
- Flexible workflow
- Granular security
- CSS support
- 508 compliance
- Personalization
- Replication
- Custom metadata
- Custom authentication
- Static site generation
- Multilanguage support

With CommonSpot Content Server you get it all. CommonSpot's exceptional blend of rapid deployment, ease of use, customization and scalability make it the leading ColdFusion content management solution.

Our rich Web publishing framework empowers developers with out-of-the-box features such as template-driven pages, custom content objects, granular security, flexible workflow and powerful ColdFusion integration hooks (just to name a few), allowing you to rapidly build and efficiently deploy dynamic, high performance Web sites.

For the past six years, PaperThin has been a leader in the ColdFusion community, and CommonSpot has been the solution of choice for organizations of all sizes, including Architect of the Capitol, AFL-CIO, Boeing, Kaiser Family Foundation, Ohio University, PGA.com and hundreds of others. CommonSpot's sophisticated feature set and affordable pricing are an unbeatable combination.

Call us today at 800.940.3087 to schedule a live demonstration of your site running under CommonSpot, or visit www.paperthin.com to learn more.

Paper Thin

You can try the test runner application by entering http://{your_server_url}/cfcunit/

If you are using the integrated Web server, {your_server_url} could stand for localhost or localhost:8500.

cfcUnit should not be installed on a live production server.

Essential Parts of the Framework

The cfcUnit distribution is made up of two main parts:
1. The core cfcUnit library components (/org/cfcunit/)
2. The Mach-II test runner application (/cfcunit/)

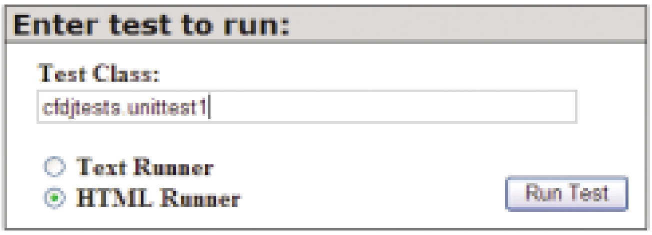
The org/cfcunit folder contains the subfolders shown in Figure 1).

Component developers are thinking in objects, so tests are modeled as objects. org\cfcunit\Object.cfc is the base class for the cfcUnit framework. All cfcUnit classes extend object.cfc.

The most important cfcUnit folder is "framework", containing the core files like Assert.cfc and TestCase.cfc.

An assertion is a Boolean expression. I suppose that many readers have already used the custom tag <cf_assert> from Hal Helms (downloadable from www.halhelms.cfm). An exemplary call could look like this:

```
<cf_assert
    assertion = "
        myAge as a: IsNumeric( |a| ); |a| GT 0; |a| LT 100 >
```



Enter test to run:

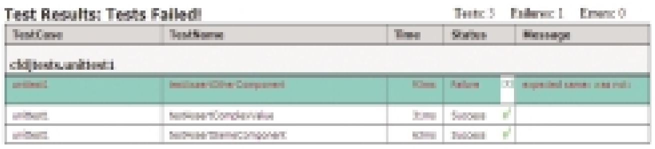
Test Class:

cfdjtests.unittest1

( ) Text Runner
[✓ HTML Runner]

Run Test

Figure 2: Test Runner form



Test Results: Tests Failed!   Tests: 3   Failures: 1   Errors: 0

TestCase	TestName	Time	Status	Message
cfdjtests.unittest1				
unittest1	[illegible]	[illegible]	Failure	[illegible]
unittest1	[illegible]	[illegible]	Success	
unittest1	[illegible]	[illegible]	Success	

Figure 3: Test results

This assertion would check for "myAge" being numeric – greater then 0 and lower then 100.

org.cfcunit.framework.Assert provides a set of assert methods like "assertTrue()" or "assertEqualsNumber()". Developers can use these asserts in their TestCases.

When an assertion fails, the framework displays the corresponding warning message.

"org.cfcunit.framework.TestCase" is one of the most important classes in the framework. The testing context is commonly referred to as "fixture". Tests in cfcUnit have a common structure – they set up a test fixture (using the method setup()), run testing code against this fixture, collect test results, and clean up the fixture (using the method tearDown()). A test case defines the test fixture to run multiple tests. TestCase contains the abstract methods runTest(), setUp(), and tearDown(). Developers who write tests could extend TestCase and override these functions.

A TestSuite (org.cfcunit.framework.TestSuite) is a composite of tests. It allows us to run many different tests together.

Sample tests to the framework files can be found in the "samples" folder, and tests that test the actual framework are found in the "tests" folder.

The base test runner in the folder "runner" is an abstract base class for all test runners. The test runner files in the folders "htmlui" and "textui" are overriding the abstract base runner class. The Text Runner has limited functionality and only reports test successes or failures. Test results are displayed in plain text (e.g., an error is displayed as "E"). The HTML Runner is used to display much more detailed and useful information.

The "extensions" folder contains useful tools like TestDecorator. The TestDecorator subclasses can add behavior before or after a test is run. After a test is run, TestResult (org.cfcunit.framework.TestResult) collects the results and distinguishes between failures and errors. Failures are anticipated; errors are unanticipated exceptions. cfcUnit checks failures through the assertions mentioned above.

Running the Sample Tests

To run the sample tests, enter one of the following class names into the form field labeled "Test Class":
- ***org.cfcunit.samples.AllTests:*** Runs all tests against the framework files
- ***org.cfcunit.samples.ArrayTest:*** Runs simple array tests
- ***org.cfcunit.samples.SimpleTest:*** Runs "assertTrue", "assertEqualsNumber", and other tests
- ***org.cfcunit.samples.money.MoneyTest:*** Runs tests against the IMoney class and subclasses

Writing Your Own Tests

Now let's write some tests. You should start writing a new TestCase by implementing a subclass of TestCase. This means that you have to create a new CFC. This component must extend the TestCase CFC included with the framework, so we use this attribute in the cfcomponent tag:

```
extends="org.cfcunit.framework.TestCase"
```

Note: This only works if "org" is directly off the root mapping.

If you need to initialize the fixture state, you should override the abstract TestCase function setUp(). You can clean up the fixture by overriding the teardown() method. For example setUp() could be used to open a network connection; the counterpart method teardown() would close this connection. Each test runs in its own fixture, so there should be no side effects among test runs.

Test methods follow the convention that they are public, have a return type of "void", have the prefix "test", and have no arguments. Test methods interact with the fixture. They verify the expected results with assertions.

The framework supports a static type way to run the test. Therefore you have to override the "runTest()" method and define the method to be invoked.

The sample file I wrote uses no static runTest() method. The framework is also capable of running tests dynamically.

To run the sample files, you have to create a folder "cfdjtests" under your webroot. Save the sample files unittest1.cfc and unittest2.cfc (available on the ***CFDJ*** Web site) into this folder. Then enter the classnames into the "Test Class" input field and hit the "Run Test" button (see Figure 2).

The first sample (see Listing 1) contains three asserts. The first one is checking for complex values. The second assert is comparing two components and is successful. The third assert will fail, because the assert "assertSameComponent" is comparing two different components – org.cfcunit.samples.money.-money and org.cfcunit.samples.money.moneybag. (Code examples for this article can be downloaded from www.sys-con.com/coldfusion/sourcec.cfm.) As you can see in Figure 3, only two asserts are successful.

The second sample (see Listing 2) demonstrates the use of the org.cfcunit.framework.TestSuite component and the addTestSuite() method. This method adds the tests from the given class "cfdjtests.unittest1" to the suite.

Conclusion

cfcUnit is a lightweight framework that makes the process of writing unit testing code easier and more efficient. Unit testing code is a good way to test individual components in isolation from other components. Unit testing can be a great help and can save you time in tracking down and fixing low-level bugs.

Resources

- *cfcUnit: Unit testing framework for ColdFusion:* www.cfcunit.org
- *OpenXCF project,* http://openxcf.sourceforge.net
- *JUnit.org: Unit testing framework for Java:* www.junit.org
- *Paul Kenny's Blog – category cfcUnit:* www.pjk.us/paul/blog/index.cfm?mode=cat&catid=7C59BA81-3048-2897-56F676CAFF026A08
- www.xprogramming.com/software.htm
- www.mach-ii.com
- http://cftestingkit.sourceforge.net
- http://c2.com/cgi/wiki?TestingFramework
- *Beck, Kent. Test-Driven Development: by Example, Addison-Wesley: Boston, 2003:* www.amazon.com/exec/obidos/ASIN/0321146530/objectmentorinc/002-7152366-8413656

"I was totally intimidated by Java, but I knew I had to learn it. Your class taught me what I honestly thought I couldn't be taught." - Sharon T

Java for ColdFusion Programmers?

Java for ColdFusion Programmers, the five-day Hal Helms, Inc. training class is designed for ColdFusion programmers; no previous Java experience is needed. You'll learn how to think in objects and program in Java.

For class information and registration, come to halhelms.com.

CFDJ Advertiser Index

ADVERTISER	URL	PHONE	PAGE
2004 READERS' CHOICE AWARDS	WWW.SYS-CON.COM	888-303-5282	45
ACTIVEPDF	WWW.ACTIVEPDF.COM		4
CF_UNDERGROUND VI	WWW.CFCONF.ORG/CF_UNDERGROUND6/	301-424-3903	19
CFDYNAMICS	WWW.CFDYNAMICS.COM	866-233-9626	11
CLEARNOVA	WWW.CLEARNOVA.COM/thinkcap		23
CFDJ	WWW.SYS-CON.COM/COLDFUSION/	888-303-5282	35
HAL HELMS, INC	WWW.HALHELMS.COM		33
HOSTMYSITE.COM	WWW.HOSTMYSITE.COM/CFDJ	877-248-4678	25
IS+S	WWW.ISSJOURNAL.COM	888-303-5282	43
INTERAKT ONLINE	WWW.INTERAKTONLINE.COM		6
INTERMEDIA.NET	WWW.INTERMEDIA.NET	800-379-7729	COVER IV
MACROMEDIA	WWW.WEBPUBLISHINGSYSTEM.COM		COVER II
MACROMEDIA MAX 2004	WWW.MACROMEDIA.COM/go/max		13
MX DEVELOPER'S JOURNAL	WWW.SYS-CON.COM/MX/SUBSCRIPTION.CFM	888-303-5282	47
PAPERTHIN	WWW.PAPERTHIN.COM	800-940-3087	31
SEAPINE SOFTWARE	WWW.SEAPINE.COM	888-683-6456	15
SECRETS OF THE JAVA MASTERS	WWW.SYS-CON.COM/FREECD	888-303-5282	34
SERVERSIDE	WWW.SERVERSIDE.NET	888-682-2544	2
SYS-CON REPRINTS	KRISTIN@SYS-CON.COM	201-802-3026	37
WEB SERVICES EDGE EAST	WWW.SYS-CON.COM/EDGE	201-802-3066	39
WEBCORE TECH	WWW.WEBCORETECH.COM	877-WCT-HOST	COVER III

- *Link, Johannes. Unit Testing in Java: How tests drive the code. Morgan Kaufmann: San Francisco, 2002.* www.amazon.com/exec/obidos/ASIN/1558608680/objectmentorinc/102-7436013-4738519

About the Author

Harry Klein is cofounder and CTO at CONTENS Software GmbH, a leading supplier of enterprise content management software. He is a Certified Advanced ColdFusion developer and Microsoft MSCE.

klein@contens.de

Listing 1: cfdjtests.unittest1 – simple test

```
<cfcomponent
 extends="org.cfcunit.framework.TestCase"
 hint="CFDJ Sample 1, simple CFCUnit Test"
 output="false">
 <!---
   current assert list:
     assertTrue
     assertFalse
     assertSimpleValue
     assertComplexValue
     assertComponent
     assertObject
     assertEqualsString
     assertEqualsNumber
     assertEqualsBoolean
     assertEqualsStruct
     assertEqualsArray
     assertSameStruct
     assertSameComponent
     assertNotSameStruct
     assertNotSameComponent
     assertNull
     assertNullComponent
     assertNotNull
     assertNotNullComponent
  --->
 <cffunction name="testAssertComplexValue" returntype="void" access="pub-
lic" output="false">
   <cfset assertComplexValue(StructNew())>
 </cffunction>
 <cffunction name="testAssertSameComponent" returntype="void"
     access="public" output="false">
   <cfset var obj1 = CreateObject("component",
"org.cfcunit.samples.money.money")>

   <cfset assertSameComponent(obj1, obj1)>
 </cffunction>
 <!--- this test will fail --->
 <cffunction name="testAssertOtherComponent" returntype="void"
     access="public" output="false">
   <cfset var obj1 = CreateObject("component",
"org.cfcunit.samples.money.money")>
   <cfset var obj2 = CreateObject("component",
"org.cfcunit.samples.money.moneybag")>
   <cfset assertSameComponent(obj1, obj2)>
 </cffunction>
</cfcomponent>
```

Listing 2: cfdjtests.unittest2

```
<cfcomponent
 extends="org.cfcunit.Object"
 hint="CFDJ Sample 2, run sample 1"
 output="false">
 <cffunction name="suite" returntype="org.cfcunit.framework.Test"
   access="public" output="false" hint="">
   <cfset var testSuite =
     newObject("org.cfcunit.framework.TestSuite").init("CFDJ Tests")>
   <cfset testSuite.addTestSuite(newObject("cfdjtests.unittest1"))>
   <!--- add more tests to this testsuite ... --->
   <cfreturn testSuite/>
 </cffunction>
</cfcomponent>
```

Download the Code...
Go to www.coldfusionjournal.com

SPECIAL: INTRODUCTION TO FLEX
COLDFUSION Developer's Journal
Flash Remoting An Alternative Approach

ColdFusion's Application Framework

–continued from page 29

Client variables are stored on disk, either in a database, the registry, or as cookies on the client's browser. Since they are not stored in shared memory, they do not need to be locked.

Application.cfm and OnRequestEnd.cfm

You may now be saying, "So, Jeff, that cfapplication tag seems pretty powerful. And the method for creating application and session variables is simple, yet elegant. But, I don't want to have this code on every single page. Is there a simpler way?" Well, I'm glad you asked. Yes, there is a simpler way. ColdFusion has a reserve filename, "Application.cfm", which helps us set our application variables or session variables in one place.

Application.cfm is a file that runs at the start of every request for a ColdFusion page. ColdFusion will search in the current directory, then the parent directory, and so-on, all the way up to the drive root until it finds a file called "Application.cfm". If it doesn't find one, then it continues processing normally. The Application.cfm file can be considered an implicit include, and if ColdFusion does find one, the code contained within Application.cfm will be executed (the same way that an included file is executed). That means the Application.cfm file is executed on every request for a ColdFusion page in that directory and below (provided none of the subdirectories has its own Application.cfm). This makes the Application.cfm ideal for things like the cfapplication tag, initializing application or session variables, and any other business logic you need to run at the beginning of every request.

If the ColdFusion server finds an Application.cfm file it will search in the same directory for an "OnRequestEnd.cfm" file. Just like the Application.cfm, this file (if found) will execute on every page request. Unlike Application.cfm, the OnRequestEnd.cfm executes after the page is finished processing, not before. OnRequestEnd.cfm does not typically get as much use as the Application.cfm, but it's good to know it exists, as it can be useful for footers or application cleanup at the end of each request.

Conclusion

When the Web started out, it wasn't being used for anything other than static content. As time went by people started using it for much more. ColdFusion's application framework represents a powerful mechanism for turning a simple Web site into a Web application. This article gave you the groundwork; now all that remains is for you to apply it to your development. Let me know how it goes.

About the Author

Jeffry Houser has been working with computers for over 20 years and in Web development for over 8 years. He owns a consulting company, and has authored three separate books on CF, most recently ColdFusion MX: The Complete Reference *(McGraw-Hill Osborne Media).*

jeff@instantcoldfusion.com

Defending ColdFusion Against ...

By Ben Forta

This column started life as a series of e-mail threads that then morphed into blog postings at www.forta.com/blog. As these points are important and need to be articulated frequently, I morphed them yet again into a column.

Enjoy.

... Java

This seems to be coming up more and more frequently – ColdFusion developers being asked to defend ColdFusion against a planned move to Java and J2EE. And so, in case you end up in this situation, this is what you need to know.

For starters, any suggestion of "we need to stop using ColdFusion because we are going to use Java" demonstrates a complete lack of understanding of what exactly ColdFusion is. So, let's start with a brief explanation of the ColdFusion–Java relationship.

Applications written in ColdFusion (as of ColdFusion MX) are pure Java. Or, expressed slightly differently, ColdFusion runs on a J2EE server (either embedded, or one of your choosing) running a Sun-verified Java application (the ColdFusion engine), executing Java bytecode (compiled from your CFML source code). In other words, CFML (the code you write) is a developer-time consideration, not a run-time consideration. There is no CFML at runtime; at runtime you are executing pure Java, no more or less so than had you written the application in straight Java. Your ColdFusion application *is* a Java application; if you deploy a ColdFusion application what you have deployed is Java. It's as simple as that.

This means that the assertion that ColdFusion and Java are somehow mutually exclusive is just flat out incorrect. But what about the CFML code you write? Isn't that ColdFusion specific and not pure Java? And isn't that an issue? I don't think so. There is an entire industry of Java add-ons out there – tools, tags, language extensions, and more – and Java shops use these (as they should; after all, why reinvent the wheel?). If your Java code leverages third-party add-ons for reporting, or back-end integration, or charting, or ... does that make your code any less Java? Nope, not at all.

Experienced developers know that starting from step one is expensive and seldom makes sense, regardless of the language and platform. Experienced developers have toolboxes at their disposal, stuff they can leverage and reuse to be as productive as possible. Experienced developers write modular applications, separating logic and processing and presentation into tiers, allowing these to evolve independently of each other, even allowing them to be inserted or removed independently.

For Java developers, one of these tools should be ColdFusion. After all, why write dozens of lines of Java code to connect to a database when a single tag can accomplish the exact same thing (likely using the same code internally)? And why write lots of code to send an SMTP message using JavaMail APIs when a single tag can do it for you (again, using those same APIs)? You can think of ColdFusion as a bunch of prewritten Java code, stuff you can use so as to hit the ground running. And that makes your app no less Java than if you had done all the work manually.

However, some may counter that CFML is proprietary, and that the fact that you need to pay for an engine to execute your code somehow makes it non-Java. I have actually heard this from customers. So is this a valid issue? Again, I don't think so. For starters, paid does not equal proprietary. After all, these same customers do not balk at spending big bucks on their J2EE servers (and management tools and professional services and more). Furthermore, there are indeed third-party CFML engines out there. I am not going to comment on how good they are and how viable an alternative they are – that's irrelevant. What is relevant is that they exist, and that means that CFML is not a single-vendor or proprietary.

Great, so ColdFusion simplifies Java development, and ColdFusion applications are no less Java than applications written in low-level Java directly. But simplicity and abstractions require sacrificing power, right? Wrong! ColdFusion applications can (and should) leverage Java; Java APIs, Java classes, JavaBeans, JSP tags, you name it, ColdFusion can leverage it because ColdFusion itself is Java. It's that simple.

So, ColdFusion or Java? The answer should be *yes*, ColdFusion is Java, and Java development can benefit from ColdFusion. This is not an either/or proposition, it's a "you can have it all so why the heck would you want to do it any other way?" proposition.

... ASP

Microsoft ASP has been an important player in Web application scripting since, well, since a year or so after the introduction of ColdFusion. From an application development process viewpoint, ColdFusion and ASP are not that different. Both are script based, both are very page centric, both embed server-side processing and client-side presentation code in source files, and both are implemented as HTTP server add-ons (ASP via ISAPI, ColdFusion via ISAPI and more). ASP and ColdFusion can coexist, and indeed, as most ColdFusion deployments are on Windows, the likelihood of ASP and ColdFusion coexisting (even if ASP is not used) is very high.

The ASP versus ColdFusion discussion used to come up regularly. But not anymore. Now that Microsoft has essentially abandoned any future development on classic ASP, replacing it with ASP.NET, few organizations are embarking on brand new ASP deployments. But having said that, if you do need to defend ColdFusion against ASP, here's what you need to know.

For starters, ASP capabilities are a subset of those of ColdFusion. Or put differently, ColdFusion can do anything that ASP can do, and a whole lot more too. The reverse is not true. Sure, ASP can be extended (using COM objects) to do just about anything that ColdFusion can do, but that's just it, you need to extend ASP – it's your responsibility to do so. Simple things that ColdFusion developers take for granted, like being able to generate an e-mail message, or process an uploaded file, or generate a business chart, none of those are native ASP functionality.

And this is not mere boasting, this is important, because it's the way to head off the "but ASP is free" argument. Sure, ASP is free for starters, but buy all the add-on bits you need to make it functionally equivalent to ColdFusion (even ColdFusion Standard, and even ColdFusion 3 or 4!) and you'll end up paying far more than ColdFusion costs. Sure, ASP is cheaper initially, but you get what you pay for. Or rather, you don't get what you don't pay for. And when you do pay for it, you'll end up paying a whole lot more.

And that's just looking at initial costs. ASP development is also far more time consuming than ColdFusion development. Even if you're comfortable in the languages used, you'll still have to write lots more code to get the job done. Even the execution of simple SQL statements is far more complex in ASP – one tag versus lots of lines of ugly code. More code = longer development time = costs more. Plus, more code = more complex ongoing maintenance = costs even more.

At the risk of sounding like an MBA, when you look at the total cost of ownership, ASP is not the cheaper option at all. Oh, and on top of all that, ASP is proprietary, a single vendor solution, and you're married to Windows boxes (no Linux, no Unix, no portability).

Maybe this is why, as already stated, most ColdFusion servers run on Windows, Windows boxes that likely already have ASP installed. Why? Because hundreds of thousands of developers have figured out that free can be far too expensive.

... ASP.NET

Comparing ASP.NET to ColdFusion is difficult. Actually, it's not just difficult, it's simply incorrect, and not an apples-to-apples comparison. In order to defend ColdFusion against a "we are moving to ASP.NET" claim, you (and whoever is involved in the decision making) need to take a big step back. Why? Simple, because ASP.NET is part of Microsoft's .NET solution, and ASP.NET apps take advantage of the .NET Framework and infrastructure, just like ColdFusion apps take advantage of J2EE. In other words, deciding between ColdFusion and ASP.NET (and indeed, defending ColdFusion against ASP.NET) first requires a .NET versus J2EE discussion.

J2EE and .NET are remarkably alike, both in terms of objectives and the range of their various components and systems. Of course, applications and application development with the two platforms are not alike at all; everything from tools to languages to methodologies are different. At their respective cores, both .NET and J2EE provide the building blocks and technologies needed to build applications. Security abstractions, database support, back-end integration, system level services, transactions and messaging, run-time services, and more are all provided by the platforms themselves. Both J2EE and .NET provide "safe" environments in which applications run (the JVM and CLR respectively); both J2EE and .NET support the use of different languages within these environments (although this potential has been realized to a greater degree in .NET); both have a scripting solution designed for Web applications (JSP or ColdFusion for J2EE, ASP.NET for .NET); and both are incredibly powerful and capable.

Many organizations are going through a J2EE or .NET discussion, usually independent of any discussion about

ColdFusion. And there are pros and cons to both options. J2EE wins when vendor independence, openness, and portability are a priority. .NET wins when it comes to tools, a better client experience, or simply a commitment to the Microsoft way (there is more to it than that, but that's an entire column unto itself).

However, as many are discovering, J2EE versus .NET is not always an either/or proposition. In fact, lots of organizations are discovering that they need both, and that the future is decidedly heterogeneous. This is especially true for larger organizations where there's room for both, and interoperability (primarily via SOAP) makes this a workable option.

If an organization has made the strategic decision to bet its future solely on Microsoft and .NET, then they probably should use ASP.NET. Sure, ColdFusion can coexist and interoperate with the .NET world, but ASP.NET will likely be the preferred option. For organizations going the J2EE route, well, I've covered that one already. But for most organizations, ColdFusion remains compelling, leveraging the worlds of J2EE natively and .NET via SOAP. In fact, some organizations have discovered that ColdFusion is the simplest way to create client applications that talk to both J2EE and .NET back ends, if that is needed.

So, ColdFusion or ASP.NET? That depends on what your IT future looks like. And unless the future is Microsoft and Windows only, ColdFusion remains an important cog in the IT engine.

... PHP

PHP is not one that comes up often; there is not a significant overlap between PHP developers and ColdFusion developers. But, in the interests of presenting the complete story, here is what you need to know.

PHP is also script based. Pages contain code that is processed by the PHP engine. The PHP engine itself is open source, and the PHP language uses a syntax borrowed from C, Java, and Perl (the latter is important, as PHP is particularly popular with former Perl developers). PHP runs on all sorts of systems, and uses functions and APIs for all sorts of processing.

PHP's stated goal (as per the official PHP FAQ) is to "allow Web developers to write dynamically generated pages quickly." Ironically, developers with both CFML and PHP experience will tell you that while PHP development may be quicker than using C or Java, it does not come close to that of ColdFusion and CFML.

There is no refuting PHP's power; PHP developers have an impressive and rich set of functions and language extensions available, and that is definitely part of PHP's appeal. But this power comes at a cost.It takes a lot more PHP code to do what simple CFML tags do, and as explained before, more code = greater cost. If you truly need that power, then PHP may indeed be an option worth considering (although CFML+Java would likely give you the same power and more). But if what you really need is to "write dynamically generated pages quickly," then PHP leaves much to be desired.

One of the most compelling arguments for PHP is its cost there is none. But, as explained earlier, the cost of software is only one part of the total cost of application development, arguably the least significant cost in the long run. It is for this reason that despite being incredibly popular with developers and consulting shops, corporate users have been slow to adopt PHP as a development platform (which in turn is why the PHP versus ColdFusion discussion comes up so infrequently).

The bottom line is if you really need the power and flexibility of PHP and can justify the longer development cycles and more code to manage and maintain it, then PHP should seriously be considered. But if time to completion and total cost of ownership are an issue, then ColdFusion wins hands down.

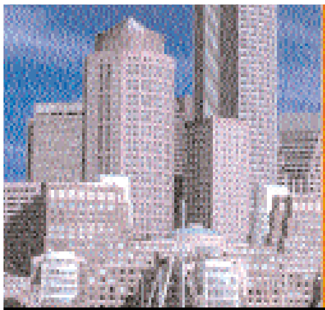
Conclusion

And there you have it, the elevator-pitch arguments needed to defend ColdFusion, if the need so arises. Of course, the ultimate defense is results, and delivering results is something ColdFusion developers have proven themselves to be incredibly good at.

About the Author

Ben Forta is Macromedia's senior product evangelist and the author of numerous books, including ColdFusion MX Web Application Construction Kit *and its sequel,* Advanced ColdFusion MX Application Development, *and is the series editor for the new "Reality ColdFusion" series. For more information visit www.forta.com.*

ben@forta.com



web services EDGE conference&expo

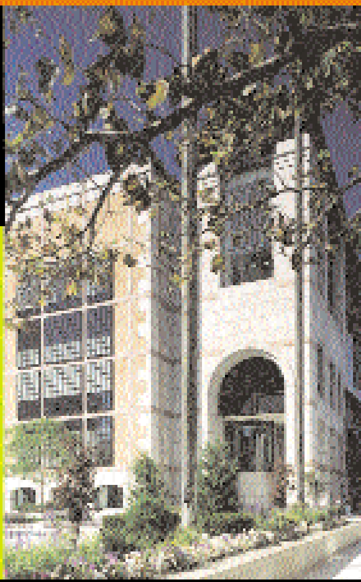
Web Services Edge 2005 East

International Web Services Conference & Expo

Announcing Web Services Edge 2005
Extending Call for Papers to August 30th
Submit your proposal today!

The Largest *i*-Technology Event of the Year!

Guaranteed Minimum Attendance 3,000 Delegates



Tuesday, 2/15: Conference & Expo | Wednesday, 2/16: Conference & Expo | Thursday, 2/17: Conference & Expo

Hynes Convention Center, Boston, MA
February 15-17, 2005

Join us in delivering the latest, freshest, and most proven Web Service solutions… at the *Fifth Annual Web Services Edge 2005 East-International Conference & Expo* as we bring together IT professionals, developers, policy makers, industry leaders and academics to share information and exchange ideas on technology trends and best practices in secure Web services and related topics including:

- Transitioning Successfully to SOA
- Federated Web services
- ebXML
- Orchestration
- Discovery
- The Business Case for SOA
- Interop & Standards
- Web Services Management
- Messaging Buses and SOA
- Enterprise Service Buses
- SOBAs (Service-Oriented Business Apps)
- Delivering ROI with SOA
- Java Web Services
- XML Web Services
- Security
- Professional Open Source
- Systems Integration
- Sarbanes-Oxley
- Grid Computing
- Business Process Management
- Web Services Choreography

3-Day Conference & Education Program

features:

- Daily Keynotes from companies building successful and secure Web Services
- Daily Keynote Panels from each Technology Track
- Over 60 sessions and seminars to choose from
- Daily Training Programs that will cover Web Service Security, J2EE, and .asp.NET
- FREE Full-Day Tutorials on .NET, J2EE, MX, and WebSphere Tutorials
- Opening Night Reception

Interested in Exhibiting, Sponsoring or Partnering?

Becoming a Web Services Edge Exhibitor, Sponsor or Partner offers you a unique opportunity to best position your organization's message and visibility to a targeted audience of Web Services Professionals. Make your plans now to reach the most qualified software developers, engineers, system architects, analysts, consultants, group leaders, and C' level management responsible for Web Services, initiatives, deployment, development and management at the regions best known IT business address, The Hynes Convention Center in Boston.
For exhibit and sponsorship information please contact Jim Hanchrow at 201.802.3066, or email at jimh@sys-con.com.

Sponsored by:

JDJ · WebServices · SD Times · wldj · .NET Journal · WebSphere Journal · Information Storage+Security Journal · XML Journal · asp.netPRO · MX · eaiIndustry

Contact for Conference Information: Jim Hanchrow, 201-802-3066, jimh@sys-con.com

SYS-CON EVENTS

www.sys-con.com/edge

ColdFusion

For more information go to...

U.S.

Alabama
Huntsville
Huntsville, AL CFUG
www.nacfug.com

Alaska
Anchorage
Alaska Macromedia User Group
www.akmmug.org

Arizona
Phoenix
www.azcfug.org

Arizona
Tucson
www.tucsoncfug.org

California
San Francisco
Bay Area CFUG
www.bacfug.net

California
Riverside
Inland Empire CFUG
www.sccfug.org

California
EL Segundo
Los Amgeles CFUG
www.sccfug.org

California
Irvine
Orange County CFUG
www.sccfug.org

California
Davis
Sacramento, CA CFUG
www.saccfug.org

California
San Jose (temporary)
Silicon Valley CFUG
www.siliconvalleycfug.com

California
San Diego
San Diego, CA CFUG
www.sdcfug.org/

California
Long Beach
Southern California CFUG
www.sccfug.org

Colorado
Denver
Denver CFUG
www.denvercfug.org/

Delaware
Kennett Square
Wilmington CFUG
www.bvcfug.org/

Delaware
Laurel
Delmarva CFUG
www.delmarva-cfug.org

Florida
Jacksonville
Jacksonville, FL CFUG
www.jaxfusion.org/

Florida
Winter Springs
Gainesville, FL CFUG
www.gisfusion.com/

Florida
Plantation
South Florida CFUG
www.cfug-sfl.org

Florida
Tallahassee
Tallahassee, FL CFUG
www.tcfug.com/

Florida
Palm Harbor
Tampa, FL CFUG
www.tbmmug.org

Georgia
Atlanta
Atlanta, GA CFUG
www.acfug.org

Illinois
East Central
East Central Illinois CFUG
www.ecicfug.org/

Indiana
Avon
Indianapolis, IN CFUG
www.hoosierfusion.com

Indiana
Mishawaka
Northern Indiana CFUG
www.ninmug.org

Iowa
Johnston
Des Moines, IA CFUG
www.hungrycow.com/cfug/

Kentucky
Louisville
Louisville, KY CFUG
www.kymug.com/

Louisiana
Lafayette
Lafayette, LA MMUG
www.cflib.org/acadiana/

Maryland
Lexington Park
California, MD CFUG
http://www.smdcfug.org

Maryland
Rockville
Maryland CFUG
www.cfug-md.org

Massachusetts
Quincy
Boston, MA CFUG
www.bostoncfug.com

Michigan
East Lansing
Mid Michigan CFUG
www.coldfusion.org/pages/index.cfm

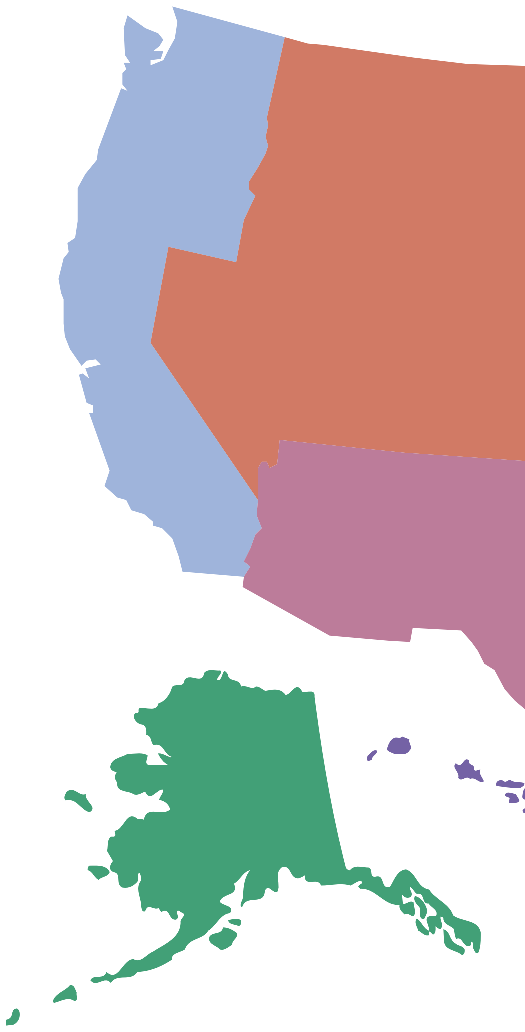
Minnesota
Brooklyn Park
Twin Cities CFUG
www.colderfusion.com

Missouri
Overland Park
Kansas City, MO CFUG
www.kcfusion.org

Missouri
O'Fallon
St. Louis, MO CFUG
www.stlmmug.com/

New Jersey
Princeton
Central New Jersey CFUG
http://www.cjcfug.us/

Nevada
Las Vegas
Las Vegas CFUG
www.sncfug.com/



New York
Albany
Albany, NY CFUG
www.anycfug.org

New York
Brooklyn
New York, NY CFUG
www.nycfug.org

New York
Syracuse
Syracuse, NY CFUG
www.cfugcny.org

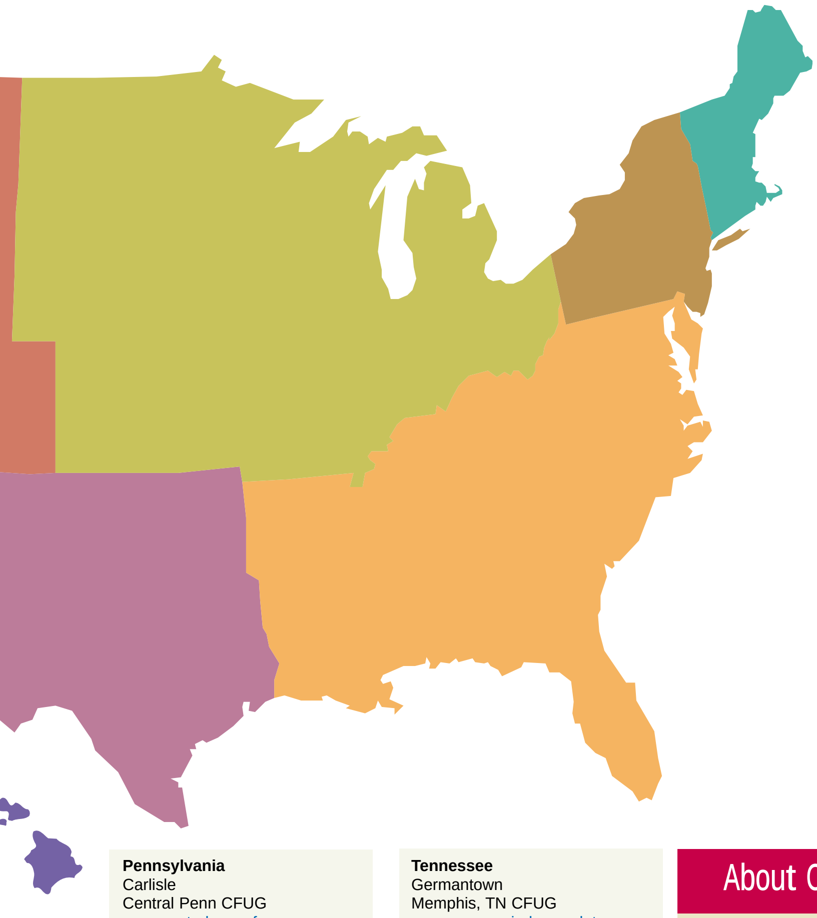
North Carolina
Raleigh
Raleigh, NC CFUG
www.ccfug.org

Ohio
Dayton
Greater Dayton CFUG
www.cfdayton.com

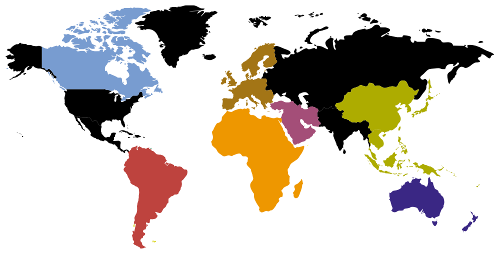
Oregon
Portland
Portland, OR CFUG
www.pdxcfug.org

User Groups

http://www.macromedia.com/cfusion/usergroups



INTERNATIONAL



Australia ACT CFUG www.actcfug.com	**Italy** Italy CFUG www.cfmentor.com
Australia Queensland CFUG www.qld.cfug.org.au/	**Japan** Japan CFUG cfusion.itfrontier.co.jp/jcfug/jcfug.cfm
Australia Southern Australia CFUG www.cfug.org.au/	**Scotland** Scottish CFUG www.scottishcfug.com
Australia Victoria CFUG www.cfcentral.com.au	**South Africa** Joe-Burg, South Africa CFUG www.mmug.co.za
Australia Western Australia CFUG www.cfugwa.com/	**South Africa** Cape Town, South Africa CFUG www.mmug.co.za
Brazil Brasilia CFUG www.cfugdf.com.br	**Spain** Spanish CFUG www.cfugspain.org
Brazil Rio de Janerio CFUG www.cfugrio.com.br/	**Switzerland** Swiss CFUG www.swisscfug.org
Brazil Sao Paulo CFUG www.cfugsp.com.br	**Thailand** Bangkok, Thailand CFUG thaicfug.tei.or.th/
Canada Kingston, ON CFUG www.kcfug.org	**Turkey** Turkey CFUG www.cftr.net
Canada Toronto, ON CFUG www.cfugtoronto.org	**United Kingdom** UK CFUG www.ukcfug.org
Ireland Dublin, Ireland CFUG www.mmug-dublin.com/	

Pennsylvania Carlisle Central Penn CFUG www.centralpenncfug.org	**Tennessee** Germantown Memphis, TN CFUG www.mmug.mind-over-data.com
Pennsylvania Exton Philadelphia, PA CFUG www.phillycfug.org/	**Texas** Austin Austin, TX CFUG www.cftexas.net/
Pennsylvania State College State College, PA CFUG www.mmug-sc.org/	**Texas** Corinth Dallas, TX CFUG www.dfwcfug.org/
Rhode Island Providence Providence, RI CFUG www.ricfug.com/www/meetings.cfm	**Texas** Houston Houston Area CFUG www.houcfug.org
Tennessee LaVergne Nashville, TN CFUG www.ncfug.com	**Utah** North Salt Lake Salt Lake City, UT CFUG www.slcfug.org

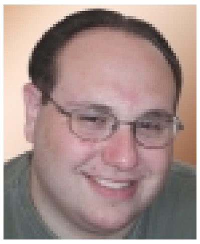
About CFUGs

ColdFusion User Groups provide a forum of support and technology to Web professionals of all levels and professions. Whether you're a designer, seasoned developer, or just starting out – ColdFusion User Groups strengthen community, increase networking, unveil the latest technology innovations, and reveal the techniques that turn novices into experts, and experts into gurus.

Using ColdFusion For Network Monitoring

Find the exact piece of data you need

By Ryan Emerle


By Matthew I. Fusfield

One of the best features of ColdFusion is its versatility. Very few other languages or environments offer the cross-platform, open framework for development that the ColdFusion platform provides.

As CF developers, we have implemented countless Web-based workflow and content management style applications. As an Internet service and hosting provider, however, we used a number of different technologies to monitor and manage our network. While a huge number of commercial and free software packages exist for managing various parts of a network, our employer lacked a single platform for managing corporate information *and* our network. Fortunately, we use ColdFusion so we're familiar with its extensibility and could take advantage of that to meet our need.

One of the key technologies used in monitoring our network was SNMP, the Simple Network Management Protocol. While the term "simple" may be relative, SNMP is an accepted standard for monitoring and managing network devices. Servers, routers, switches, UPSs, environment sensors, and nearly any other kind of network-connected device can often "speak" SNMP to communicate information about their operation and status.

SNMP works by sending messages, called protocol data units (PDUs), to different parts of a network. SNMP-compliant devices, called agents, store data about themselves in Management Information Bases (MIBs) and return this data to the SNMP requesters.

A popular application among network administrators is MRTG. MRTG is an open source application that generates useful graphs of LAN and WAN utilization. Simply put, MRTG reads information from a router via SNMP, processes it, and generates a graph of the information that the router reports about bandwidth utilization. This is just one example of what SNMP can be used for. Because of SNMP, MRTG can work with devices from multiple vendors and even graph information from two different types of routers side by side.

This summary of SNMP is fairly simple, but is enough to give you the basic idea of what it can do.

CFX_SNMP

For ColdFusion developers without CFQUERY, ODBC (or JDBC more recently) would be much more difficult to implement. Unfortunately, ColdFusion does not have a similar tag for communicating via SNMP. This was corrected with the development of the CFX_SNMP, which is available for purchase and download at www.insapi.com/. CFX_SNMP allows ColdFusion applications to retrieve data from network devices through the use of SNMP, and is written in Java for cross-platform compatibility.

Use of the tag is as follows:

```
<cfx_snmp
host="router.host.com"
community="public"
oid="1.3.6.1.2.1.2.2.1.2"
r_qResults="results">
```

The HOST attribute simply specifies the host name or IP address to connect to. This can be a router, a server, network switch, or any other piece of SNMP-enabled network equipment. The COMMUNITY attribute specifies the SNMP community string to use, which is similar to a password for accessing the SNMP device. Next, the OID (object id) attribute is a numeric identifier. The object ID selects the specific piece of information to be retrieved. SNMP organizes objects in a tree fashion, similar to a file system, which can be "walked". The example above indicates that we should first select object 1, then the third subobject, then the sixth subobject, and so on, until we retrieve the specific data needed. Finally, the R_QRESULTS attribute specifies the name of the query to return data. This query can be accessed as any other ColdFusion query, using Query-of-Query, CFOUTPUT, or CFLOOP.

The result set will have four columns: the OID, the data type, a hexadecimal representation of the data, and a string representation of the data.

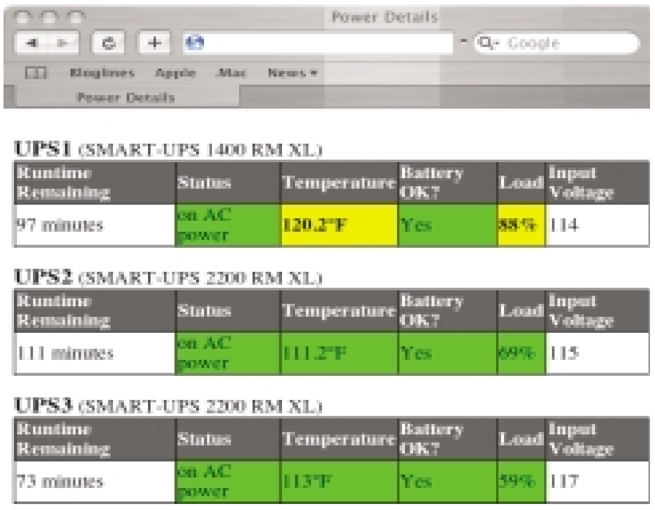
There are a few other useful features of the tag. If you specify an OID that contains multiple values ("subobjects"), the resulting query will contain rows for each. Further, the OID attribute of the tag will accept a comma-separated list of OIDs, and will return the values of each in a separate row of the query.

Example Application – UPS Power Monitoring

One problem we've experienced in our data center is monitoring the status of the power equipment. Each rack has at least one APC UPS battery backup unit, as well as an APC MasterSwitch for remote control of power. While each has an easy-to-use Web and text Telnet interface, in an outage, opening multiple browser windows to track your power utilization is not very efficient. For this reason, we used the CFX_SNMP to create a power "dashboard" application that displays the status of each UPS and MasterSwitch. This allows a network operator to determine how much runtime is available during a power failure, the load placed on each UPS, and even the internal temperature of each UPS.

The ups.cfm script (see Listing 1) has a number of declarations used throughout. First, a variable called COMMUNITY is set to the community string used. While there is no requirement that all network devices share a community string, management is made easier if all devices share a common community string. A struct called UPS contains named keys for each OID that will be referenced later. The sample script assumes that there are multiple UPSs being monitored, named in DNS as ups1, ups2, and ups3. This can be changed by altering the list in the first CFLOOP tag.

For each UPS looped over, the CFX_SNMP tag executes with the seven OIDs passed and returns a query called STATS. This query has seven rows, one for each OID in the same order as passed to CFX_SNMP (row 1 has the first OID, row 2 has the second OID, and so on). Each row contains OID, Hex, Type, and Value columns. The value column is the most important, as it contains the values that we will need to process and output. OID is the object ID being returned, and Type is a string representing the datatype of the data in the value column. Finally, the hex column is the hexadecimal value returned by the network device. You can add a <CFDUMP VAR="#stats#"> tag to



UPS1 (SMART-UPS 1400 RM XL)

Runtime Remaining	Status	Temperature	Battery OK?	Load	Input Voltage
97 minutes	on AC power	120.2°F	Yes	88%	114

UPS2 (SMART-UPS 2200 RM XL)

Runtime Remaining	Status	Temperature	Battery OK?	Load	Input Voltage
111 minutes	on AC power	111.2°F	Yes	69%	115

UPS3 (SMART-UPS 2200 RM XL)

Runtime Remaining	Status	Temperature	Battery OK?	Load	Input Voltage
73 minutes	on AC power	113°F	Yes	59%	117

Figure 1: Sample script

see the full output of the tag.

The script outputs a table for each UPS (see Figure 1), with column headers for runtime, status, temperature, battery status, power load, and input voltage. Part of the title for each table is the UPS name and model number. Since we specified the OID for model number six in our OID list for CFX_SNMP, the sixth row of the query will contain the string representation of the UPS model number. This value is accessible as #stats.value[6]#.

The script then outputs each row of the query. The first OID in the list is the runtime remaining of the UPS. This is the maximum amount of time before the batteries will run out if utility power is cut. The UPS expresses this in "timeticks" (100th of 1 second), which we convert to minutes using the Evaluate() function. The next OID is the UPS system status, which indicates if the system is running on utility power, battery, or some other specific condition (such as a low-voltage boost). The UPS returns this represented as a number, so we use a CFSWITCH to select the appropriate, operator-friendly text to display. Third is the UPS operating temperature. This is returned in Celsius and has been converted to Fahrenheit scale in this script for display using the Evaluate() function. We have also added some conditional logic to select the color of the display; if the UPS temperature begins to reach the maximum operating temperature, the display turns yellow, and then red once it exceeds the recommended maximums. The next OID is a simple binary value of whether the UPS battery needs to be replaced. Again, some logic has been added to change the color so an operator can know at a glance if there's a problem. The fifth row in the query contains information about the percentage load on the UPS. As in the other rows, some conditional logic has been added for display purposes. Finally, we output row 7 of the query, which is the input line voltage to the UPS. This is often useful if a backup generator powers the UPS.

Putting It All Together

This is a very simple example of using SNMP from within ColdFusion. Since the CFX_SNMP tag returns a standard ColdFusion query object, anything that you can do with a query, you can do with SNMP data. SNMP devices could easily be SOAP-enabled by writing a few CFCs with the appropriate calls to CFX_SNMP. The tag could easily be used with a database for logging historical information. This data can then be summarized or graphed with CFCHART to produce network status or outage reports.

The most difficult part for most ColdFusion developers will be determining the OIDs with the data needed. The OIDs in this example are for APC UPSs, but what if you want to monitor router usage or server disk space? SNMP will allow you to do this, but figuring out the correct OIDs to query is often a challenge. For this reason, a number of utilities exist that allow you to traverse, or "walk" the SNMP tree to find the exact piece of data you need.

Getif (www.wtcs.org/snmp4tpc/getif.htm) is one such freeware utility for Windows. Using Getif and a large MIB library (also available on the same site) you can easily browse for a particular value on nearly any kind of network equipment. Once you have found the needed OID, simply copy it into your CFML code and use the live data however needed.

ColdFusion is a great environment for creating business applications; integrating it with common hosting and networking systems can give organizations a unified look into their operations. In addition to databases, LDAP directories, mail servers, and FTP servers, ColdFusion can now access the wide array of network information via SNMP.

About the Authors

In addition to building enterprise applications in ColdFusion, Ryan Emerle has a total of 10 years' experience with Java, C/C++, and other high-level languages in varying environments.

Matthew I. Fusfield has over 10 years' experience in the Internet industry. He is a certified ColdFusion developer and has used it in the creation of dozens of systems and public Web sites.

rde@emerle.net
matt@fus.net

Listing 1

```
<html>
 <head>
   <title>Power Details</title>

   <!---
   UPS Monitoring Sample Script
    Ryan D Emerle (rde@emerle.net) and Matthew I Fusfield (matt@fus.net) --
->

   <style>
     td { border: solid 1px black; }
     table {
         border-collapse:collapse;
          border: solid 1px black;
       }

   </style>

   <!--- refresh the screen every 10 seconds --->
   <cfoutput>
   <META HTTP-EQUIV="Refresh" CONTENT="10;URL=#cgi.script_name#">
   </cfoutput>
 <head>
<body>


<!--- set your SNMP community name here --->
```

```
<cfset community="public">

<!--- these are the OIDs we will reference later --->
<cfset ups=structNew()>
<cfset ups["status"]="1.3.6.1.4.1.318.1.1.1.4.1.1.0">
<cfset ups["capacity"]="1.3.6.1.4.1.318.1.1.1.2.2.1.0">
<cfset ups["temp"]="1.3.6.1.4.1.318.1.1.1.2.2.2.0"> <!--- Deg celsius --->
<cfset ups["runtime"]="1.3.6.1.4.1.318.1.1.1.2.2.3.0"> <!--- Timeticks are
100ths of a second --->
<cfset ups["replace"]="1.3.6.1.4.1.318.1.1.1.2.2.4.0"> <!---  1=OK,
2=Replace --->
<cfset ups["load"]="1.3.6.1.4.1.318.1.1.1.4.2.3">
<cfset ups["model"]="1.3.6.1.4.1.318.1.1.1.1.1.1">
<cfset ups["involts"]="1.3.6.1.4.1.318.1.1.1.3.2.1">


 <!--- this script assumes your UPS's are named ups1.company.net,
ups2.company.net, and ups3.company.net
     loop through this list of UPS's to monitor --->
 <cfloop list="1,2,3" index="i">

   <!---
   This produces a 7 row query (one for each OID)
   Columns: Hex, OID, Type, Value
   ---->
   <cfx_snmp
     timeout=1
     community="#community#"
     host="ups#i#.company.net"

oid="#ups['runtime']#,#ups['status']#,#ups['temp']#,#ups['replace']#,#ups[
'load']#,#ups['model']#,#ups['involts']#"
     r_qResults="stats">

   <cfoutput>
   <br>

   <big><b>UPS#i#</b></big> (#stats.value[6]#) <!--- this will display
the model number (row and OID 6) --->
   <table width="500" cellspacing=0 bordercolor="black">
     <tr bgcolor="555555">
       <td><b style="color: white">Runtime Remaining</b></td>
       <td><b style="color: white">Status</b></td>
       <td><b style="color: white">Temperature</b></td>
       <td><b style="color: white">Battery OK?</b></td>
       <td><b style="color: white">Load</b></td>
       <td><b style="color: white">Input Voltage</b></td>
     </tr>
     <tr>

          <!--- the first OID and row is runtime remaining expressed in
"timeticks" Convert into minutes and output --->

            <td>#evaluate(stats.value[1]/100/60)# minutes</td>


          <!--- the second row is the status as reported by the UPS --->
            <!--- output a text description of the UPS status --->
            <cfswitch expression="#stats.value[2]#">
                    <cfcase value="1">
                              <td bgcolor="red"><b>unknown</b></td>
                    </cfcase>
                    <cfcase value="2">
                              <td bgcolor="lime">on AC power</td>
                    </cfcase>
                    <cfcase value="3">
                              <td bgcolor="red"><b>on battery</b></td>
                    </cfcase>
                    <cfcase value="4">
                              <td bgcolor="yellow"><b>on smart
boost</b></td>
                    </cfcase>
                    <cfcase value="5">
                              <td bgcolor="yellow"><b>timed
sleeping</b></td>
                    </cfcase>
                    <cfcase value="6">
                              <td bgcolor="yellow"><b>software
bypass</b></td>
                    </cfcase>
                    <cfcase value="7">
                              <td bgcolor="yellow"><b>off</b></td>
                    </cfcase>
                    <cfcase value="8">
                              <td bgcolor="yellow"><b>rebooting</b></td>
                    </cfcase>
                    <cfcase value="9">
                              <td bgcolor="yellow"><b>switchedBypass
</b></td>
                    </cfcase>
                    <cfcase value="10">
                              <td
bgcolor="yellow"><b>hardwareFailureBypass </b></td>
                    </cfcase>
                    <cfcase value="11">
                              <td
bgcolor="yellow"><b>sleepingUntilPowerReturn </b></td>
                    </cfcase>
                    <cfcase value="12">
                              <td bgcolor="yellow"><b>onSmartTrim
</b></td></td>
                    </cfcase>
            </cfswitch>

          <!--- the third row is the UPS temperature. Convert into
degrees F and change the color if we
               are getting close or exceed the recommended operating
temperatures --->

            <cfset temp=evaluate((stats.value[3] * 1.8) + 32)>
            <cfif temp gte 145>
                    <td bgcolor="red"><b>#temp#&deg;F</b></td>
            <cfelseif temp gte 120>
                    <td bgcolor="yellow"><b>#temp#&deg;F</b></td>
            <cfelse>
                    <td bgcolor="lime">#temp#&deg;F</td>
            </cfif>

          <!---- the forth OID specifies if the UPS thinks the battery
needs to be replaced --->
          <cfif stats.value[4] eq 1>
                    <td bgcolor="lime">Yes</td>
            <cfelse>
                    <td bgcolor="red"><b>No</b></td>
            </cfif>


          <!--- the fifth OID is the percentage load on the UPS. Again,
change color depending on how
             close to 100% we get --->

            <cfif stats.value[5] gte 90>
                    <td bgcolor="red"><b>#stats.value[5]#%</b></td>
            <cfelseif stats.value[5] gte 80>
                    <td bgcolor="yellow"><b>#stats.value[5]#%</b></td>
            <cfelse>
                    <td bgcolor="lime">#stats.value[5]#%</td>
            </cfif>


          <!--- the sixth row is used above to specify the model number

          The seventh row is the line-in voltage --->

            <td>#stats.value[7]#</td>

      </tr>
    </table>
    </cfoutput>
  </cfloop>


</body>
</html>
```

Download the Code...
Go to www.coldfusionjournal.com

2004 RCAs
Readers' Choice Awards
Nominations Open
"The Oscars of the Software Industry"
ColdFusion Developer's Journal CFDJ Readers' Choice Award
Nominate the Best of the Best:
Tell us what tools, solutions, services, and books deserve recognition.
Cast your vote at www.SYS-CON.com

'Selling' ColdFusion

How to confront the naysayers

By Hal Helms

"I'm getting a lot of resistance," my client told me. "Prospective clients love the functionality of our application, but they shy away when they hear that it's developed in ColdFusion. Their IT people don't like it and we're starting to lose sales."

I've heard this from quite a few development shops that have asked me for help in "selling" ColdFusion. The most common line of defense, suggested by some at Macromedia, is that "ColdFusion is Java." My client had heard this and asked me for my advice on adopting this strategy.

"I don't like it," I told them. "It's an argument that won't fly with Java developers. Yes, it's true that ColdFusion compiles down to bytecode, but that's a different proposition from saying that ColdFusion is Java – and Java developers know that. You lose a lot of credibility with them for proposing that."

My client continued: "We're being asked for UML class diagrams more often now and I'm starting to think we may have to port the entire application to Java or C#. Would you recommend that?"

"I don't like that either," I told them. "Porting an entire application is time-consuming, expensive, and very risky: there's no guarantee that your first cut will be successful."

"Well, you're not bringing me any good news," the client said ruefully. "Or did you just save a bunch of money on car insurance?"

If you've found yourself in my client's position, you may very well be asking yourself the same questions they did. Do we stay with ColdFusion? Do we try to migrate to another platform? In this article, I propose a strategy that I have found to be successful. First, though, let's examine the situation from the prospective customer's IT department. What is their concern over ColdFusion? Although the arguments given usually involve scalability and security, I believe that their discomfort is simply that ColdFusion doesn't have the security blanket that wraps Java and .NET. The old saying was that "Nobody ever got fired for buying IBM." Today we might update that: "Nobody ever got fired for going with Java or .NET." ColdFusion just seems too risky.

Given that their fears are not strictly rational – good ColdFusion applications can be very secure and scalable – countering their given arguments in a tit-for-tat fashion is probably not going to be very successful. Also, admittedly, technologies such as Java do provide greater capabilities and flexibility than an all-ColdFusion solution. That is to say, their fears are not strictly irrational. Those of us who have found ColdFusion to be a highly effective way to develop applications find ourselves in a tricky position. Let's go back to the discussion with my client.

"What if you were to show your client that your application is built on the foundation of a Model-View-Controller (MVC) architecture? All IT shops will know – and respect – that design pattern. Using MVC, the components of an application are separated into one of three categories: model components handle the business logic and data persistence of the application, the "heavy lifting", if you will; the view components provide the user interface to the client; the controller components mediate communication between the model and view components."

"Yes," my client told me. "Several of the shops asked if we were using MVC, but don't we have to adopt something like the Java Struts framework for that?"

"Not at all," I said. "The two most popular frameworks in the ColdFusion world – Fusebox and Mach-II – are both solid implementations of MVC."

"Well, that's good. But that doesn't solve my problem, or rather, my client's problem with ColdFusion."

"Well, let's examine that," I said. "To clients, the user interface – the View in Model-View-Controller – is the application, but to IT folks, that's just eye candy. To them, the application is the business logic – the model. And there's no reason we need to write those model components in ColdFusion. In fact, writing them in Java can bring us a lot of benefits, as well as meeting the concerns of your prospects' IT shops."

"How so?"

"Java is a language that's well adapted to dealing with complexity. And well-designed Java code can be designed so that applications can evolve gracefully. It's one of the great benefits of objec-oriented (OO) technology. The rigor inherent in Java with its strong data typing can assist us greatly in spotting errors before the application is deployed."

"But Java…" my client said. "Isn't that really hard? And my developers don't know Java."

"Well, in answer to your first question, no, Java isn't really hard. The syntax for Java can be learned very quickly. What's hard about Java is what's hard about any language: learning to develop a good architecture using the language. You can write the type of code you're doing now – procedural code – in Java and, in fact, that's what a lot of so-called Java developers do. But doing that won't get you any real benefits. To benefit from Java or C#, you need to understand what makes object-oriented development so powerful in order to apply this to your applications."

"Are there books that explain that?"

"There are many excellent books, but trying to learn OO from a book is a frustrating experience for most people. It's a bit like trying to learn Japanese from a book. I think you'd do better getting help from an experienced OO teacher/mentor. You also want to learn how to use UML to produce class diagrams to describe the architecture of your model."

"Which is what my prospects' IT shops want to see."

"Yes, but you won't be producing them simply to jump through hoops set up by your prospect. Those UML class diagrams are how your developers plan their architecture. And the choice of that architecture can be and should be influenced by a study of design patterns."

"That's a topic that also comes up with the IT department," my client said. "I don't really understand them."

"Design patterns are nothing more than time-tested solutions to recurring design problems. They offer help to developers in coming up with flexible solutions. An example might be a situation where you have to represent salaried workers and hourly workers in your model component."

"So, you'd have a Worker class that would be subclassed to HourlyWorker and SalariedWorker. Right?"

"But what if you decide to change an employee from hourly to salaried?"

"Oh. And changing the classes somehow; would that work?"

"That's possible, but problematic. It would be better to implement a solution that allows you to swap out pay algorithms. That's exactly what the Strategy design pattern allows you to do. You define an abstract PayStrategy that's subclassed to SalariedPayStrategy and HourlyPayStrategy. Then your Worker class specifies that it holds a PayStrategy, but it doesn't matter which particular subclass it holds. This lets you swap Hourly for Salaried and vice versa. That's a solution that works very well, but which isn't immediately obvious; it takes some experience to get to that solution. And that's exactly what design patterns give you: a tested solution that provides flexibility."

"So, I have to rewrite all my model code in Java?"

"Not immediately. By adopting an MVC framework such as Mach-II or Fusebox, you can wrap your existing code into the appropriate components and slowly begin adopting Java in the model, using appropriate design patterns."

"Is that possible?"

"Yes. ColdFusion makes it quite easy to connect to Java objects. So once your developers begin learning Java, they can gradually adopt it within the application. It's a far less risky strategy than a complete rewrite and you get the best of both worlds: Java for the business logic and ColdFusion for the presentation logic. Your application becomes more robust and you satisfy the concerns of your prospects."

With the next release of ColdFusion (dubbed "Blackstone"), powerful new presentation and report-writing capabilities will be within the reach of ColdFusion programmers. By beginning the process of learning and integrating Java, developers can continue to provide great applications while answering the technology concerns of prospective customers.

About the Author

Hal Helms (www.halhelms.com) is a Team Macromedia member who provides both on-site and remote training in ColdFusion, Java, and Fusebox. Hal is cofounder of the Mach-II project.
hal.helms@teamallaire.com

Top 10 Web Security Tips

You can't be too careful...

By Michael Smith

All data-driven Web sites are potentially vulnerable to hackers, whether they are written in ColdFusion or another language. Fortunately, a few simple steps can make your site much more secure.

1. Have an Error Handler

By default ColdFusion displays a detailed error message when there is a problem in your code. Yes, I know – your code has no bugs in it – but what if a hacker deliberately creates an error by manipulating URL parameters or faking out a form submit? If they can make an error and view the error message, they may see your data source and table names, which could help them create a SQL injection attack (see Tip 6 for more on this). To protect your error messages from prying eyes, add a CFERROR tag in your application.cfm to catch all errors and display a harmless message to users and hackers alike. I also like to e-mail myself a private copy of the error message so that I can either fix it or be warned that a hacker is on my site.

In your application.cfm use:

```
<CFERROR type="EXCEPTION" template="error_exception.cfm"
mailto="michael@teratech.com">
```

And in your error handler file error_exception.cfm, use this code:

```
<CFMAIL to="#error.MailTo#"
   from="info@teratech.com"
   subject="ColdFusion Error in #error.Template#">
         Address: #error.RemoteAddress#
         Template: #error.Template#
         Date: #error.DateTime#
         Message: #error.Diagnostics#
</CFMAIL>
```

To be extra careful, trap for the REQUEST type for CFERROR after the EXCEPTION type. This is a backup error handler for sites with high user interface requirements. You might also turn off the setting for showing the full path to error messages in the CF Admin. Neither should you give more information in error messages than you have to. For example, on a logon screen don't say that "the user id is correct but password wrong," because that gives extra information. Just say "logon denied."

2. Prevent Cross-Site Scripting

Cross-site scripting occurs when a hacker changes your URL, FORM, or COOKIE parameters to either create an error or to view unauthorized information. Changing URL parameters is just a matter of editing the URL line in the browser. Changing hidden form fields is a bit harder. Here is how the hackers do it: they save the HTML of your posting page on their server, edit the hidden form fields, edit the FORM ACTION parameter to point to the submit page on your server, and then submit their copy of the posting page from their browser.

In the past this technique has been used to change the price for an online store item from $100 to $1, causing great losses for the Web site. (Of course, it is bad security design to put such data in the form anyway – the less you expose the better.) Fake form submits can also bypass any client-side validation that you are doing on the form fields submitted. What to do? Protect your URL and hidden FORM variables from outside manipulation.

For URL variables you can use the free functions URLEncrypt and URLDecrypt from CFLib (www.cflib.org is a library of ColdFusion functions that you can use in your code). URLEncrypt will let you display your URL variables as one big coded string (like the big URL strings you can see on Amazon.com) and decrypt them on the receiving page back into individual URL variables.

For Form fields, use a Checksum field so you can prevent them from being changed.

In the calling page:

```
<input type="hidden" name="ID" value ="#GetBouquets.BouquetID#">
<input type="hidden" name ="checksum" value
="#hash(GetBouquets.BouquetID)#">
```

In the submit page:

```
<CFIF hash(FORM.ID) neq FORM.checksum>
   Bad Form<CFABORT>
</CFIF>
```

The hash() function gives a check sum for a string so if the string is changed, the hash of it will change too. For multiple hidden fields you can first combine them into one string, then hash that. It doesn't matter how long the string is, the hash function always returns 32 characters.

I also suggest that you validate all URL and FORM parameters with the CFPARAM tag. This tag cannot only provide a default value, but it can also check data types. If you want to make sure that a URL parameter is supplied, just leave out the DEFAULT parameter of CFPARAM and an error will be thrown if the URL variable is missing:

```
<CFPARAM NAME = "URL.ID"  type = "numeric">
```

3. Remove Dangerous Characters

Some characters can be used by hackers to inject dangerous SQL, HTML, or JavaScript into your site, for example a SCRIPT tag or an onClick event. You can easily prevent these kinds of problems by scanning and removing dangerous characters from all URL and FORM variables in your application.cfm using the REReplaceNoCase() function. Typical characters to remove include: "(", ")", "<", ">", "/", and "|". The SafeText function from CFLib is also useful when you want to allow some HTML but not the dangerous stuff.

4. Prevent Fake Form Submits

As I mentioned in Tip 2, hackers can submit fake forms from their server to your submit pages. To prevent this, check the referrer Web page using the CGI variable CGI. HTTP_REFERER. If it is in the same domain as your site, then all is okay. You can test for this either in each submit page or once in your application.cfm file like this:

```
<CFIF not (CGI.HTTP_REFERER contains
"http://www.mysite.com/")>
   <CFABORT>
</CFIF>
```

Note: Because the CGI.HTTP_REFERER comes from the browser it can be hacked, so the above test will not stop the determined hacker, only the casual one!

It is also a good idea to protect CFINCLUDE and CFMODULE files from being run stand-alone; they may do bad things or generate error messages when run this way. One easy method is to use a common file naming or subdirectory convention for all includes and modules. In the application.cfm test the CGI.script_name to see if it contains this string. The CGI.script_name variable gives you the URL path and filename of the current CFM file being run.

This method is especially important when coding using the Fusebox methodology, because the site structure and certain filenames will be known without seeing your source code. In a Fusebox application only the index.cfm file needs to be run. Here is how you can protect the includes files in this case:

In Application.cfm:

```
<CFIF CGI.SCRIPT_NAME contains "index.cfm">
   <!--- ok to run --->
<CFELSE>
         <CFABORT SHOWERROR="Protected
page">
</CFIF>
```

Alternatively, place included files or cfmodule files outside the Web root.

5. Stop Unauthorized Data Mining

As I mentioned in Tip 2, hackers can change the values of URL or FORM parameters to view other people's data. For example, if you have a parameter user ID=5 in your page's URL variables, they might try running the page with user ID=4 to see what data they can see.

Again, the fix is to prevent users from changing sensitive parameters by using checksums or encryption. However, don't get carried away and protect all parameters, because some may be used by other sites linking to you. A classic example is the book ISBN on Amazon's site: most of their URL is one long encrypted string, but the ISBN is left in plain text so that other sites can link to books on Amazon's site.

6. Validate Parameters and Prevent SQL Injection Attacks

One way hackers can hurt your site is by running bad SQL in your queries by typing the extra SQL into URL or FORM parameters that you use in your CFQUERY statements. A little-known feature of SQL Server is that you can run two SQL statements from one CFQUERY by separating them with a semicolon. Hackers exploit this by injecting extra SQL via URL parameters.

For example, suppose you have a CFQUERY in page.cfm that contains the following SQL:

```
SELECT * FROM EMP WHERE ID = #URL.USER ID#
```

And normally the page is run like this:

```
http://myserver/page.cfm?USER ID=7
```

This will cause the following SQL to be run in your CFQUERY:

```
SELECT * FROM EMP WHERE ID=7
```

Now the hacker adds extra SQL commands on the end of the URL variable (the %3B and %20 encode a semicolon and a space, respectively, in a URL string):

```
http://myserver/page.cfm?USER
ID=7%3BDELETE%20FROM%20CustomerTable
```

This will cause the following SQL to be run in your CFQUERY:

```
SELECT * FROM EMP WHERE ID=7 ;DELETE FROM
CustomerTable
```

This will run both SQL commands and delete all your records from the table CustomerTable!

Other attacks use the vertical bar character (|) to run VBA functions in Access. One of the most dangerous VBA functions is shell(), which will run any command string in a shell on your server. Just think what a FORMAT command might do to your server! In SQL Server the similar xp_cmdshell function is equally dangerous and can be turned off in the SQL Server admin.

To prevent SQL injection hacking, use <CFQUERYPARAM> on all SQL parameters. If you haven't come across this tag before, it is used to validate the data types of SQL parameters inside CFQUERY tags. Here is an example to make things clearer:

```
<CFQUERY name="getFirst" DATA SOURCE="cfsnip-
pets">
SELECT * FROM EMP WHERE ID =
<CFQUERYPARAM value="# URL.USER ID #"
CFSQLType="CF_SQL_integer"> </CFQUERY>
```

In this code the URL.USER ID will only be allowed to be an integer; any hanky-panky with semicolons or bad extra SQL will generate an error (that you can catch, of course). A side benefit of using CFQUERYPARAM is that your queries will run faster because SQL Server or Oracle will be more likely to use a cached query plan instead of recalculating the query plan from scratch.

Note: Some programmers use the ColdFusion val() function to protect numeric parameters, but CFQUERYPARAM also works with other datatypes and – unlike val – it generates an error when the data type is incorrect. Also, the previous tips on removing dangerous characters and encrypting URL variables help to prevent SQL injection attacks, but these should be used only in addition to CFQUERYPARAM, and not instead of it.

7. Use Server-Side Validation to Back Up Client-Side Validation

As I mentioned in Tip 2 on cross-site scripting, fake form submits can be used to remove or modify any client-side validation that you use on your site. While I recommend using client-side validation (such as CFFORM, JavaScript, and _ field name validation) to make the user interface better, I do not rely on it for security. For that purpose I use server-side validation to back up client-side validation. For example, if you have a field that must be a date, use this code on your submit page:

```
<CFIF isdefined(“FORM.MyDate”)>
<CFIF not isdate(FORM.MyDate)>
   Bad date <CFABORT>
</CFIF>
   </CFIF>
```

Alternatively, you could use the CFPARAM statement:

```
<CFPARAM NAME=”FORM.MyDate” TYPE=”date”>
```

8. Harden Your Logon Code

You may have password-protected areas of your site for admin or members only. Be sure that the “front door” of your logon screen is strong. For example, require hard passwords from your users, such as ones with more than eight characters and requiring numbers as well as letters. The PasswordCheck function at CFLib can help you with this task by verifying the length and type of characters in a password. And if you need an easy-to-type random password, check out MakePassword at CFLib. It will create a password with random characters without the confusion between lowercase L and the number 1, etc.

Once all your users have strong passwords, consider making them change them every six months to further protect your login. And don’t store passwords in your database, in case a hacker gets a copy of your user database. Instead, store the hash of the password instead of plain text by using the hash() function. Here is how to use this in your logon screen:

```
<CFQUERY NAME="CheckPerson" DATA
SOURCE="UserData">
SELECT PasswordHash FROM SecureData WHERE user
ID=
<CFQUERYPARAM VALUE="#user ID#"
CFSQLType="CF_SQL_CHARVAR"> </CFQUERY>
<CFIF Hash(Form.Password) IS NOT
CheckPerson.PasswordHash>
          <CFLOCATION
URL="unauthenticated.cfm">
<CFELSE>
          <CFLOCATION
URL="authenticated.cfm">
</CFIF>
```

For a really secure login, consider timing out the login for an hour for a particular user ID after three failures at logging in. This prevents hackers from automatically trying thousands of common passwords via an automated submit program.

9. Prevent Timeout Client/Session Backdoors

When you are using client or session variables to protect secure areas of your site there is still the problem of your users leaving their machines logged in while they go to lunch. A hacker could see the open browser and steal confidential data. For this reason it is wise to use a short timeout of 10 minutes.

Normally I handle this in my code rather than changing how long the session or client variables exist. Partly because it is easy for me to control and customize for each user, but also so that while I log out users automatically, I can still offer them the chance to reenter their password and not lose any data that is stored in their session.

I would also suggest that you make the associated session/client cookies CFID and CFTOKEN automatically delete after the browser closes. Otherwise a hacker may open the browser and use the history feature to go to your secure pages if the user forgot to log out! Here is the code to make CFID and CFTOKEN expire immediately on browser close.

```
<CFCOOKIE expires="NOW" name="CFID"
value="#cookie.cfid#">
<CFCOOKIE expires="NOW" name="CFTOKEN"
value="#cookie.cftoken#">
```

This keeps the current cookie names and values but makes sure that they disappear when the browser closes.

10. Avoid Trojan Horse Uploads

Some sites let you upload graphics or other files. If you are not careful, hackers can use this feature to upload program files and then run them from your Web site. Or they might try to overwrite critical system files by using ../ paths. To prevent this, always store uploaded files outside your Web root and strip out any path information from filenames. Also be careful if you use CFCONTENT to display a file where the filename is given via a URL parameter. The page can be misused to send back your source code if a hacker gives the path and name of your CFM files instead of a graphic. Again, storing displayed files outside of the Web root helps, as does validating that the parameter filename does not contain any path such as “../”.

What Security Means

Security is difficult because a hacker needs only one window to be open to get in, whereas you must close all the holes. Assume bad things will happen and code for them! Security is a way of thinking: “How can they get into this page?” Finally, realize that more knowledge is power; don’t keep security tips secret! The more people who know how to secure their sites the fewer hackers can get in.

About the Author

Michael Smith is president and founder of TeraTech, a 15-year-old Rockville, Maryland-based consulting company that specializes in ColdFusion development and training. Michael has been programming for over 25 years and has been coding in ColdFusion since version 1.5.

michael@teratech.com

One damn good Cold Fusion hosting firm!

Webcore Technologies specializes in ColdFusion, ASP and SQL hosting. We offer these solutions in both dedicated server and shared virtual environments. Webcore can design and implement clustered or load-balanced solutions, managed firewalls, VPN's, multi-site failover, and more. In addition, we also offer corporate Internet access, web site design, and technology consulting services.

Our in-house tier 1 level data center enables us to provide the most reliable hosting in the industry. Our Microsoft and Cisco certified team ensures that all support issues are resolved promptly and professionally. Unlike other hosting companies that answer with a phone attendant, you will receive a live person when you call Webcore. No phone attendant, no frustrations, just world class customer service and support the first time you call.

Cisco Systems

Microsoft CERTIFIED Partner

Webcore TECHNOLOGIES

Webcore Technologies, Inc.
877.WCT.HOST
www.webcoretech.com

be the pilot!

FREE SETUP on Shared Hosting Accounts With ColdFusion MX Support
Use Promo Code CFDJ2004

INTERMEDIA.NET

WE DARE YOU TO TAKE A FREE TEST FLIGHT!

Managing technology that runs your business is a matter of trust and control. INTERMEDIA.NET gives you both.

TRUST. Since 1995 we have been providing outstanding hosting service and technology to our clients. Don't take our word for it... take theirs.

"The support and service that you offer are nothing short of golden. The high quality of your system and service for CF customers is something one could only ever dream of." – Claude Raiola, Director, AustralianAccommodation.com Pty. Ltd.

CONTROL. We give you instant control over your site, server and account configuration changes. No more submitting requests and waiting for someone else to take action. You are in control to pilot your business through its daily needs.

BE THE PILOT. Take a free test flight and see what our HostPilot™ Control Panel offers you beyond all others. Check out our SLA guarantees. To see more testimonials and to find out about our competitive advantages, visit our Web site at www.Intermedia.NET.

Managed Hosting • Shared Hosting • Microsoft Exchange Hosting

Call us at: 1.800.379.7729 • Visit us at: WWW.INTERMEDIA.NET

HostPilot™