

Using Remoting with Flex and ColdFusion CFC

Part 1 – Using the ColdFusion Flash Remoting Gateway

Abstract

This document describes the basic requirements to use Flex with ColdFusion's Remoting gateway and explains how to use named and unnamed RemoteObjects.

Intended Audience

You should be familiar with Flex and ColdFusion in general. Especially, you should have an understanding of the ColdFusion CFC concept. Also, you should have an idea of what Flash Remoting is all about.

Requirements

- Flex 1.5 (either installed as standalone or deployed into a servlet container)
- ColdFusion MX 7 (either installed as standalone or deployed into a servlet container)
- A code editor

For simplicity:

- Flex root directory will be referred to as [FlexRoot]
- ColdFusion root directory will be referred to as [CFRoot]
- Flex server is assumed to run on <http://flexserver/flex>
- ColdFusion server is assumed to run on <http://cfserver/cfusion>

The examples have **not** been tested with a “merged” installation of CF and Flex – I never merged both applications as I don't think it's a good thing to do.

Introduction

With the release of Flex 1.5 and ColdFusion MX 7 people often ask how to connect both worlds. Using the Flash Remoting connectivity between the Flex frontend and the CF backend seems quite obvious, but unfortunately there's only few information available on this specific topic, especially nothing official. For this reason I decided to write up a few pages that should get you into the right direction. This guide may not be complete but it shows a setup that works fine in our projects.

A Simple EchoService.cfc

As a quick start we'll just create a very simple CFC that acts as an echo service – it will just return the input argument back to our Flex application.

- Inside [CFRoot] create a new folder called `remoting`
- Inside [CFRoot]\remoting create a new folder called `samples`
- Inside [CFRoot]\remoting\samples create a new file called `EchoService.cfc`
- Open `EchoService.cfc` and edit it

Listing 1 – EchoService.cfc

```
<cfcomponent displayname="EchoService">
  <cffunction name="echo" access="remote" returntype="string">
    <cfargument name="inputString" type="string" required="true">
    <cfreturn arguments.inputString>
  </cffunction>
</cfcomponent>
```

Nothing special about this CFC so far so let's invoke this thing!

Invoking a CFC – The Basics

Two things are very important to understand:

- 1) if you want to invoke a CFC from your Flex application, you **cannot** use the Remoting gateway of your Flex server. You **have** to use the one of your ColdFusion server.
- 2) because of 1) you need a **crossdomain.xml** file on your ColdFusion server if Flex and ColdFusion are hosted on different domains.
 - For a detailed discussion about the crossdomain.xml and the Flash Player security model see: http://www.macromedia.com/devnet/mx/flash/articles/fplayer_security_print.html

There are three different techniques to tell Flex to use a different Remoting gateway than the included one.

Technique 1: Setting the endpoint attribute of the RemoteObject tag

The easiest way to let your Flex application use a different Remoting gateway is to set the `endpoint` attribute of the `mx:RemoteObject` tag to the URL of ColdFusion's Remoting gateway. By default, this URL is:

<http://cfserver/cfusion/flashservices/gateway>

If you open your CF server's Remoting gateway URL inside a browser your CF server will redirect your request to the server's start page. That's because the above URL is rather a **mapping** to a J2EE servlet than a URL that points to a real directory.

By default, CF is configured to route requests to URLs that contain a specific pattern (in this case `/flashservices/gateway/*`) to the servlet that handles incoming and outgoing Flash Remoting traffic. Simple HTTP Requests made by browsers aren't accepted as Flash Remoting requests and therefore get redirected.

The URL pattern and the servlet mapping is defined (and can be changed) in the file

[CFServer] \WEB-INF\web.xml

Figure 1 – URL Pattern definition in web.xml

```
256     <servlet-mapping id="macromedia_mapping_1">
257         <servlet-name>FlashGateway</servlet-name>
258         <url-pattern>/flashservices/gateway/*</url-pattern>
259     </servlet-mapping>
```

Now that we know the URL of CF's Remoting gateway we are able to write a simple Flex application that invokes our EchoService.cfc

- Inside [FlexRoot] create a new folder called `remoting`
- Inside [FlexRoot]\remoting create a new folder called `samples`
- Inside [FlexRoot]\remoting\samples create a new file called `EchoExample01.mxml`
- Open `EchoExample01.mxml` and edit it

Listing 2 – EchoExample01.mxml

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
  xmlns:mx="http://www.macromedia.com/2003/mxml">

  <mx:Script>
  <![CDATA[

    private function doClick():Void {
      ro.echo(input.text)
    }

    private function doResult(event:Object):Void {
      output.text = event.result;
    }

    private function doFault(event:Object):Void {
      alert("Error invoking CFC: " + event.fault.faultstring);
    }
  ]]>
</mx:Script>

  <mx:RemoteObject
    id="ro"
    endpoint="http://cfserver/cfusion/flashservices/gateway"
    source="remoting.samples.EchoService"
    result="doResult(event) "
    fault="doFault(event) "
  />

  <mx:Panel title="EchoService">
    <mx:Form defaultButton="sendButton">
      <mx:FormItem label="Input:" direction="horizontal">
        <mx:TextInput id="input"/>
        <mx:Button id="sendButton" label="Send" click="doClick()"/>
      </mx:FormItem>
      <mx:FormItem label="Output:">
        <mx:TextInput id="output" editable="false"/>
      </mx:FormItem>
    </mx:Form>
  </mx:Panel>

</mx:Application>
```

The source attribute of the RemoteObject tag points to the “fully qualified class name” of the CFC – i.e. the directory path from the CF web root down to the CFC itself, separated by dots. Note that it's `remoting.samples.EchoService` and **not** `remoting.samples.EchoService.cfc`!

The RemoteObject instance is given an instance name of `ro` which is used inside the `doClick()` method to invoke the remote `echo()` function of the CFC.

Ready, Set, Go? – No!

Before you try to compile and test your Flex application keep in mind that Flex no longer uses the remoting proxy which allows to access data sources from different domains. When overriding the endpoint URL you possibly need a **crossdomain.xml** file (see above) on your CF server.

If all is set, browse <http://flexserver/flex/remoting/samples/EchoExample01.mxml>

Technique 2: Overriding the default gateway by using query parameters

Another technique that's especially helpful during development is to use query parameters (FlashVars also work) to change the endpoint URL.

However, this is disabled by default and you need to enable it by changing the `<allow-url-override>` setting of the `<remote-objects>` node inside the `[FlexRoot]}\WEB-INF\flex\flex-config.xml` file.

Figure 2 – Allow URL overriding in flex-config.xml

```
<!-- use proxy specified via flashvars or query parameter ?remoteURL=XXX -->
<allow-url-override>true</allow-url-override>
```

Using this setting, overriding the gateway URL is as simple as <http://flexserver/flex/remoting/samples/EchoExample01.mxml?remoteURL=http://anotherserver/flashservices/gateway>

This technique is of course **not recommended** in production mode and is mentioned here only for the sake of completeness! Make sure to revert to the default setting (`<allow-url-override>>false</allow-url-override>`) before carrying on!

Technique 3: Using named RemoteObjects

This is probably the best way to use RemoteObjects: instead of hardwiring the `source` and `endpoint` attributes in the MXML file (which exposes implementation details and increases maintenance) the RemoteObjects are defined on the hosting server. Inside the MXML code you merely refer to a named RemoteObject by using its **unique name**.

BTW: prior to ColdFusion MX 7 it wasn't possible to use named RemoteObjects with the CF Remoting gateway at all. This became possible because the Remoting gateway in CF MX 7 is (almost?) identically to the one used in Flex and can be configured externally.

So let's add a named RemoteObject definition to CF's Remoting gateway!

- Open `[CFRoot]\WEB-INF\gateway-config.xml`
- Locate and modify the `<whitelist>` node
- Add a named object definition for `remoting.samples.EchoService`

Listing 3 – Named RemoteObject definition

```
<whitelist>
  <unnamed>
    <source></source>
  </unnamed>
  <named>
    <object name="Echo">
      <source>remoting.samples.EchoService</source>
    </object>
  </named>
</whitelist>
```

Note that the default `<source>*<source>` setting has been replaced by a more strict one – otherwise **any** CFC could potentially be accessed through Flash Remoting! The `<source/>` node is also surrounded by a `<unnamed/>` tag to distinguish between named and unnamed RemoteObjects.

The new `<named/>` node defines the named RemoteObject. The `name` attribute sets up the name by which this object will be available from the Flex application (“Echo”). The `source` node simply is the fully qualified class name (see above) of the CFC.

Important: you’ll have to **restart** your CF server for the changes to take effect!

Now that the CF server is setup, you still need to tell the Flex server to use the CF server’s Remoting gateway. Instead of using the `endpoint` attribute (or the query param technique) you can also **globally** change the gateway location. This affects **all** Flex applications on this Flex server instance so be sure you don’t mess up your installation!

- Open `[FlexRoot]\WEB-INF\flex\flex-config.xml`
- Locate and modify the `<amf-gateway>` setting

Listing 4 – Setting the amf-gateway to CF server Remoting gateway

```
<!-- The location of the AMF Gateway. The value below is used
when calling a page over http or when protocol is specified as http -->
<amf-gateway>http://cfserver/cfusion/flashservices/gateway</amf-gateway>
```

Important: Due to a bug in Flex 1.5 you also need to add a dummy named RemoteObject entry in the Flex server’s RemoteObject whitelist. If you don’t do that, your MXML **won’t compile**.

Simply add a named RemoteObject definition with a name of “Echo” to the remote-objects whitelist. The `source` can point to anything – it’s just there for the compiler.

Listing 5 – Adding a dummy named RemoteObject entry

```
<!-- a list of urls accessible via this remote objects proxy -->
<whitelist>
  <!-- whitelist config for unnamed objects -->
  <unnamed>
    <source></source>
    <!--
    For security, the whitelist is locked down by default.

    Uncomment the source element below to enable access to all classes
    during development.

    We strongly recommend not allowing access to all source files
    in production, since this exposes Java and Flex system classes.
    <source>*</source>
    -->
  </unnamed>
  <named>
    <object name="Echo">
      <source>foo.bar</source>
    </object>
  </named>
  ...
</whitelist>
```

Important: you'll have to **restart** your Flex server for the changes to take effect!

After both servers are setup, we need to change our Flex application.

- Open [FlexRoot]\remoting\samples\EchoExample01.mxml
- Save it as [FlexRoot]\remoting\samples\EchoExample02.mxml
- Modify the RemoteObject tag to use a named RemoteObject

Listing 4 – Modified RemoteObject tag in EchoExample02.mxml

```
<mx:RemoteObject
  id="ro"
  named="Echo"
  result="doResult(event) "
  fault="doFault(event) "
/>
```

Browse <http://flexserver/flex/remoting/samples/EchoExample02.mxml>

Summary

This guide explained how to invoke a simple CFC by either using unnamed and named RemoteObjects.

The next part of this guide will show you how to pass complex data between Flex and ColdFusion and some more best practices.

Future parts will be posted at: <http://www.richinternet.de/blog>

For questions, suggestions and comments: deismann@richinternet.de