


# MAX 2006 Beyond Boundaries



Flex 2.0 and ColdFusion  
Mike Nimer  
Tapper, Nimer and Associates  
Inc.

© 2006 Adobe Systems Incorporated. All Rights Reserved.

## What is Flex?

- Flex is the name for a family of products
  - Flex Data Services (enterprise server)
  - Flex SDK (flash components and swf compiler)
  - FlexBuilder (IDE)
- Flex is the **presentation layer!**
  - Think of it as a replacement for HTML, not ColdFusion.

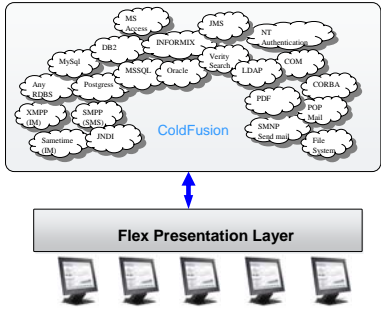
© 2006 Adobe Systems Incorporated. All Rights Reserved.

## Why does it matter to you (a Web/CF developer)

- More options for your apps
  - Old Days
    - HTML / DHTML
  - Now!
    - Flash / Flex**
    - Ajax
    - HTML
- Better separation of presentation and business logic.
- A better experience for your clients
  - Less screen reloads.
  - Richer feedback.

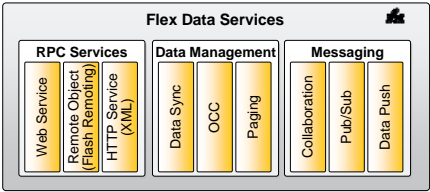
© 2006 Adobe Systems Incorporated. All Rights Reserved.

## ColdFusion as the Backend "Bridge".



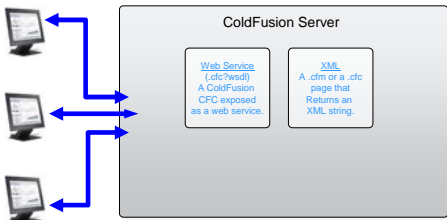
© 2006 Adobe Systems Incorporated. All Rights Reserved.

## The Flex Data Services



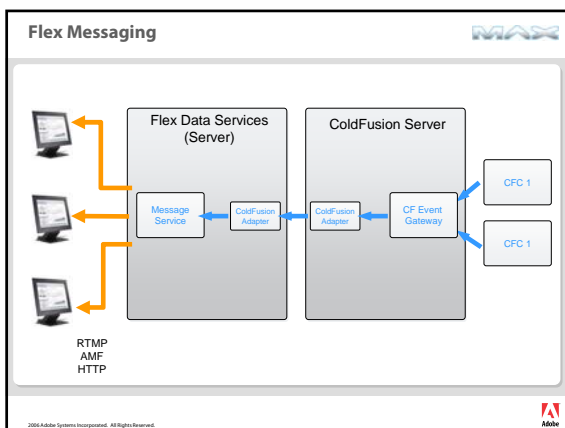
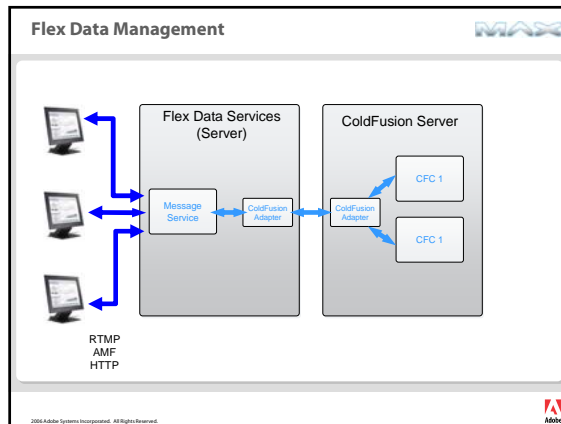
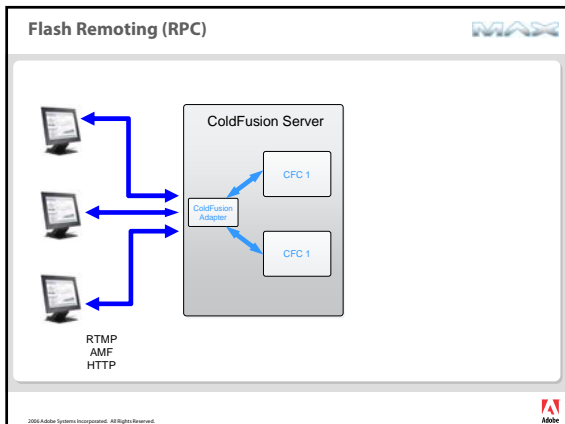
© 2006 Adobe Systems Incorporated. All Rights Reserved.

## Simple ways to talk to a server (RPC)



RTMP  
AMF  
HTTP

© 2006 Adobe Systems Incorporated. All Rights Reserved.



- ### What tools do you need.
- ColdFusion 7.0.2 (mystic)
  - A ColdFusion IDE
    - CFclipse (recommended)
      - <http://www.cfclipse.org>
    - HomeSite
    - Dreamweaver
  - FlexBuilder IDE
  - The ColdFusion plug-ins for FlexBuilder
    - Optional, but you really want them.
  - (optional) Flex Data Server
    - Required, if you want to use FDS or Messaging
    - **Not required** for RPC Services.
      - Flash Remoting
      - Web Services
      - XML
- 2006 Adobe Systems Incorporated. All Rights Reserved.

- ### Types of applications – What are you going to build?
- **Hybrid**
    - A mix of HTML and Flash movies
      - For instance a flash stock ticker running in an html page.
      - HTML wrapper (navigation) around one or more flash movies.
  - **Full App**
    - 1 big app
      - Like a Client desktop application running in a browser.
- 2006 Adobe Systems Incorporated. All Rights Reserved.

- ### First
- Forget about the standard HTML/CF Request and Response model (page reloading)
  - Time to think with events!
    - Small self-contained widgets that throw an event when something happens.
    - The widgets don't know what to do, the movie that is listening for an event does.
    - To use a Product List as an example.
      - When a product is selected the widget will yell "Bob Dylan CD was selected".
      - Whoever is listening for the product list to yell will do something, not the product list. Such as show an edit form, add to cart, or display concert tickets.
      - Allows for easy to build and maintain re-usable widgets.
- 2006 Adobe Systems Incorporated. All Rights Reserved.

Simple Example

### What do we need to build?

- We need a CFC that returns data.
  - The access attribute needs to be "remote"
- We need a flex movie that uses the <mx:RemoteObject> tag.
- We need a way to display the results.

### Example

- The CFC that returns data.

```
<cfcomponent>
<cffunction name="getQuery" access="remote" returntype="query">
<cfset var qData = "">

<cfquery name="qData" datasource="cfcodeexplorer">
select artistid, firstname, lastname, email
from artists
order by lastname
</cfquery>

<cfreturn qData />
</cffunction>
</cfcomponent>
```

### Example

- The mx:RemoteObject tag
  - Defines the CFC to call in Flash

```
<mx:RemoteObject
id="cfdata"
destination="ColdFusion"
source="simple.components.Simple">
<mx:method
name="getQuery"
result="getQuery_result(event)" />
</mx:RemoteObject>
```

### Example

- Call ColdFusion from flex

```
public function initApp():void
{
// call the CFC method
cfdata.getQuery();
}

• A function to handle the result and to populate the grid with the result from ColdFusion.
```

```
public function getQuery_result(event:ResultEvent):void
{
// populate the grid with the query returned from CF.
cfgrid.dataProvider = event.result as ArrayCollection;
}
```

### Recap

- You can return "any" type of ColdFusion variable to a Flash movie (query, array, struct, date, etc...)
- All requests are asynchronous. No more coding for the Request and Response (page reload) model.

Example 2

### OOP / Value Objects

- The flex2 Flash Remoting adapter allows you to pass object "instances" back and forth. So instead of array, structs, and queries. You can pass, Users, Products, and Setting objects (cfcs).
- Objects are defined as "CFC instances" on the server and "AS Classes" in Flex.
- Only PUBLIC properties are sent back and forth.
- Methods are not sent.
- The conversation between AS Class and CFC instance is automatic (if you code it right).

### More Complex Interaction

- OOP / Value Object

```

    graph LR
      subgraph Client [Flex/Flash movie]
        A[AS Class Book.as]
      end
      subgraph Server [ColdFusion Server]
        B[CFC Bean Book.cfc]
      end
      Adapter[Flash Remoting Adapter]
      A <--> Adapter
      Adapter <--> B
  
```

### More Complex Interaction

- OOP / Value Object
  - AS Class
    - The "OBJECT" on the client (in flex)
  - Bean type of cfc
    - The "OBJECT" on the server (in CF)
  - DAO type of cfc
    - The "Queries"
  - Gateway type of cfc
    - Utility Methods
    - I like to put all of my "remote" methods in here too.
- Note: If you use a framework for your ColdFusion applications you may create a different set of cfc's. It depends on the rules of your chosen framework.*

### Example

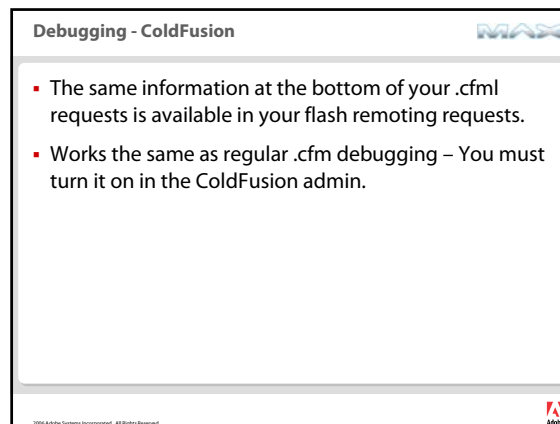
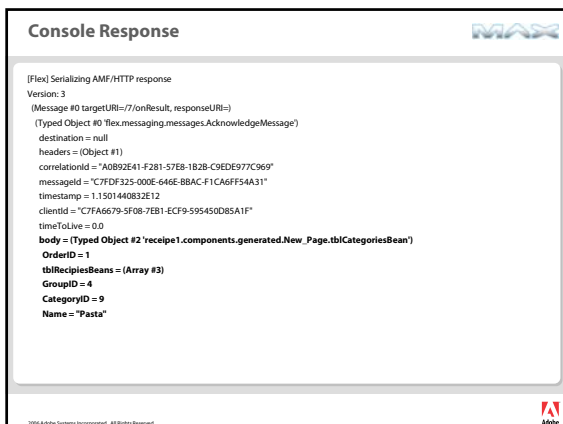
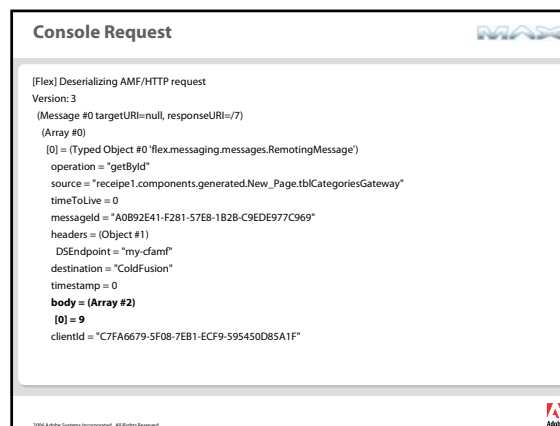
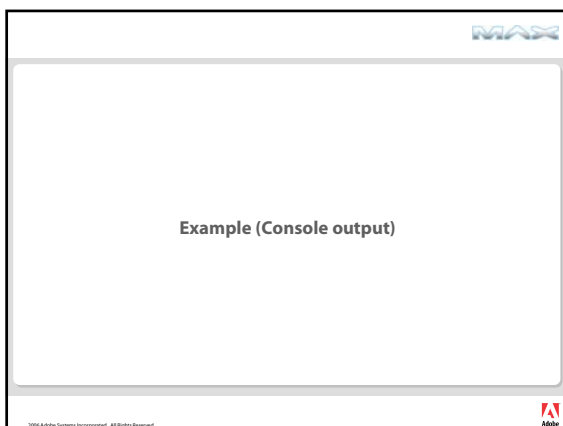
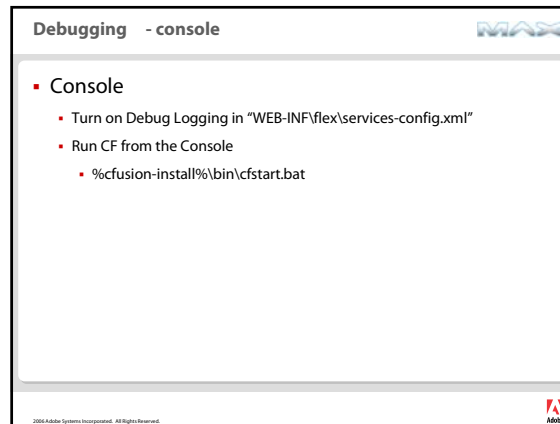
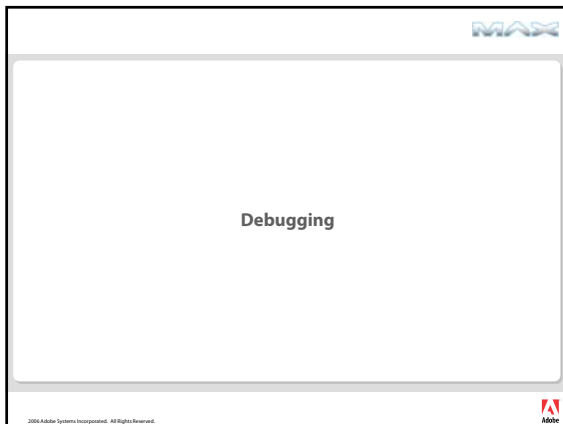
- Create the CFCs
  - There is a new "alias" attribute for <CFCOMPONENT> to help.
- Create the matching AS Class
  - You need to link the AS class to the CFC alias with AS metadata
  - [RemoteClass(alias="dot.path.to.the.cfc")]

### Binding to the "object"

- In your edit form you will want to bind the form fields back to the object. So you can send the "object" back to the server to save the values.
- Create an xml version of the Object to bind to.
  - You would send back to the server the (this.editArtist) object.

```

    <mx:TitleWindow
    xmlns:mx="http://www.adobe.com/2006/mxml"
    xmlns:model="simple"
    width="400" height="300">
    <model:Artists id="editArtist">
      <model:FIRSTNAME>{this.firstname.text}</model:FIRSTNAME>
      <model:LASTNAME>{this.lastname.text}</model:LASTNAME>
      <model:EMAIL>{this.email.text}</model:EMAIL>
    </model:Artists>
  
```



**Example**

### Configuration Details:

- Project Settings**
  - Any project that wishes to talk to ColdFusion must add this compiler property. (adjust to match your path)
    - services "C:\cfusionmx7\wwwroot\WEB-INF\flex\services-config.xml"
  - The FlexBuilder "ColdFusion project type" will set this automatically for you.
- Inline Destinations**
  - Destination and Source attribute of mxRemoteObject tag.
    - Advance Tip: Instead of linking to the services-config.xml file you can create your own "ChannelSet" object to define the url to ColdFusion.*
- Services-config.xml**
  - Destinations (the cf: you want to talk to)
  - XML Config file.
    - "WEB-INF/lib/services-config.xml"
    - Define specific destinations and the **one** cf: that destination can access
    - Must create 1 destination for each CFC, or use the "\*" wildcard to allow the inline source attribute.
    - "\*" means any CFC and the source must be defined in the <mxRemoteObject> tag.

### Sample Destination

```

<destination id="SampleDestination">
  <channel>
    <channel ref="my-of-aaef"/>
  </channel>
  <properties>
    <!-- <source> </source> -->
    <source>com.mikenimer.components.Product</source>
    <!-- define the resolution rules and access level of the cf: being invoked -->
    <access>
      <!--
      One the ColdFusion mappings to find CFCs.
      by default only CFC files under your webroot can be found. -->
      <use-mappings>true</use-mappings>
      <!-- allow "public and remote" or just "remote" methods to be invoked -->
      <method-access-level>remote</method-access-level>
    </access>
    <property>name
      <!-- cf: property name -->
      <force-cf:lowercase>false</force-cf:lowercase>
      <!-- Query column name -->
      <force-query-lowercase>false</force-query-lowercase>
      <!-- struct key -->
      <force-struct-lowercase>false</force-struct-lowercase>
    </property>name
  </properties>
</destination>
  
```

### Debugging – XML config file

- Also used to turn of different levels of logging
  - Error
  - Warning
  - Debug

```

<logging>
  <target class="flex.messaging.log.ConsoleTarget" level="Debug">
    <properties>
      <prefix>[Flex] </prefix>
      <includeDate>false</includeDate>
      <includeTime>false</includeTime>
      <includeEvent>false</includeEvent>
      <includeCategory>false</includeCategory>
    </properties>
    <filters>
      <pattern>Endpoint.*</pattern>
      <pattern>Service.*</pattern>
      <pattern>Configuration.*</pattern>
      <pattern>Message.*</pattern>
    </filters>
  </target>
</logging>
  
```

### Community

- Components**
  - Flex debug panel and a few others.
    - <http://www.mikenimer.com>
- Tools**
  - Service Capture
    - <http://kevinlangdon.com/serviceCapture/>
- Flex Frameworks**
  - "Cairngorm"
    - [http://www.iterationtwo.com/open\\_source\\_cairngorm.html](http://www.iterationtwo.com/open_source_cairngorm.html)
- Mailing Lists**
  - Yahoo FlexCoders group

**QA**