

Contents

- [BlazeDS 3.2 integration](#)
- [Overall setup of BlazeDS](#)
- [The RailoAdapter](#)
- [Using BlazeDS with Railo Express](#)
- [Using BlazeDS with Railo Express Server](#)
- [Using BlazeDS with Railo Resin Server](#)
- [Using BlazeDS with the railo.war](#)
- [Running multiple Railo Webs with BlazeDS](#)

BlazeDS 3.2 integration

Railo 3.x comes with BlazeDS 3.2, the Adobe opensource amf engine to communicate from a Flex application to a Java backend. Using AMF has a lot of advantages over the usage of REST or SOAP, if you need more information about BlazeDS please take a look at the documentation on <http://opensource.adobe.com/wiki/display/blazeds/Developer+Documentation>.

The following is a short explanation about how to use BlazeDS with Railo. First we start with an overall description about how to setup BlazeDS and how to test that it works, later down we will look at the different Railo versions (Express, Server etc) and where you can find the individual pieces of the BlazeDS setup for your Railo of choice. This text is based on Railo 3.1.2, but should apply also to future versions of Railo 3.x. It could change slightly for Railo 4 though.

Overall setup of BlazeDS

The different versions of Railo come with slightly different preparations of the following, but to get an overview we start with a standard BlazeDS/Java setup. You can test this without Railo, just download the blazeds.war file from <http://opensource.adobe.com/wiki/display/blazeds/Downloads>, unzip it and copy it on the java server of your choice. For example with tomcat you would put the unzipped blazeds directory into the webapps folder. In the following examples we will call this directory the /app-root directory.

To be able to use BlazeDS, we need to run the MessageBroker Servlet on the java server. This can be achieved by putting the following code into the /app-root/WEB-INF/web.xml file:

```
MessageBrokerServlet
flex.messaging.MessageBrokerServlet

    services.configuration.file
    /WEB-INF/flex/services-config.xml
```

1

```
MessageBrokerServlet
/messagebroker/*
```

The default place where BlazeDS looks up its configuration is in the /app-root/WEB-INF/flex directory, where BlazeDS loads the services-config.xml. In this file the endpoints are defined that can be used in the ChannelSet of the Flex application as the endpoint url. For example in a standard setup like you can find it in the blazeds.war the AMFEndpoint you would have following url

```
http://{server.name}:{server.port}/{context.root}/messagebroker/amf
```

To test all this, you need of course also the BlazeDS jars on your classpath. For example in the blazeds.war you can find the jars in the /app-root/WEB-INF/lib directory. If you restart you server and browse to <http://yourdomain/yourcontext/messagebroker/amf> you should get a white page, if you get an error message, this would mean that either you server is not running, the

MessageBroker Servlet is not running or you used different urls. :)

So far, so good, now let's take a look at what's specific for Railo to work with BlazeDS.

The RailoAdapter

The remoting communication of BlazeDS is implemented in the `flex.messaging.services.remoting.adapters.JavaAdapter` and this approach above works fine if you want to call a single java class on your backend, but because Railo is more than just a single java class, Railo provides it's own implementation of the adapter which is the `railo.runtime.net.flex.RailoAdapter`.

To call a Railo cfc from Flex you need the following: (In a Railo Express all this is already setup for you)

Open the `/app-root/WEB-INF/flex/services-config.xml` and add the following channel definition in the `<channel>` tag:

```
...
  id="my-cfamf" class="mx.messaging.channels.AMFChannel">
    uri="http://{server.name}:{server.port}{context.root}/flex2gateway/" class="flex.messaging.endpoints.AMFEndpoint"/>
    
    
  
```

Now open the `/app-root/WEB-INF/flex/remoting-config.xml` and add the following code:

```
  id="cf-object" class="railo.runtime.net.flex.RailoAdapter" default="true"/>
  ...

  ref="my-cfamf"/>
  ...

  id="ColdFusion">
    source>*/source>
```

And last but not least, in your `/app-root/WEB-INF/web.xml` you need to add the following servlet mapping

```
MessageBrokerServlet
/flex2gateway/*
```

If you now browse to <http://yourdomain/yourcontext/flex2gateway> you should have the same white page as above. Of course, to have this work, you need the `RailoAdapter` on your classpath, which means that you need to copy the `railo.jar` in your `libs` directory. To test this you don't need to setup Railo completely, just adding the jar will already makes the example work, but without Railo it's quite pointless. That's why we will look now at the three different Railo versions and where you can find these settings after you have downloaded a new Railo version.

Using BlazeDS with Railo Express

The Express version of Railo has BlazeDS already setup to be used. But we will go through the different places where you can find the pieces of code explained above.

After you downloaded a Railo Express and unzip it, you need to start it up once, so that that Railo creates the `WEB-INF` directory in the webroot folder. You can now verify that the

MessageBroker Servlet is properly started by browsing to <http://localhost:8888/flex2gateway> which should return the already familiar white page.

So it looks like everything is where we have learned above, but if you look into the WEB-INF directory, you will notice that there is no web.xml. Instead open /railo-express/etc/webdefault.xml. If you scroll down a little you will see that the MessageBrokerServlet is there as well the three mappings for /flex2gateway/*, /flashservices/gateway/* and /messagebroker/*. To be backward compatible, OpenAMF, the opensource implementation of AMF0 is still there and you can request it over /openamf/gateway/*

That's it, if you look into the /railo-express/webroot/WEB-INF/flex directory, you will find the services-config.xml along with the remotng-config.xml as you would expect. In the services-config.xml you will find four channels setup (my-cfamf, classic-cfamf, cf-polling-amf and my-cfamf-secure). Look into the remotng-config.xml and you will see that the RailoAdapter is set to default true, the my-cfamf channel is the default channel and the ColdFusion destination is there, ready to be requested by the RemoteObject from a Flex application.

Using BlazeDS with Railo Express Server

After starting the Railo Express Server version and browsing to <http://localhost:8080/flex2gateway> you will face a 404 Not Found error message. This is because by default BlazeDS is not yet setup on this version.

The services-config.xml and the remotng-config.xml are setup as you would expect and you can refer to the explanations above, but the MessageBroker Servlet is not yet setup. You need to add the following code:

```
MessageBrokerServlet
flex.messaging.MessageBrokerServlet

    services.configuration.file
    /WEB-INF/flex/services-config.xml
```

1

```
MessageBrokerServlet
/flex2gateway/*
```

You can do this in two places, either add it to the /railo-resin-express/conf/app-default.xml or add a web.xml into the /railo-resin-express/webroot/WEB-INF directory and add the servlet there.

After restarting the server you can browse to <http://localhost:8080/flex2gateway> which should return the expected white page again, without error this time.

Acutally, the missing servlet can perfectly be classified as a bug and shuld be corrected in future releases. :)

Using BlazeDS with Railo Resin Server

If you work with the full server version, go to /railo-resin/conf/app-default.xml to verify that the MessageBroker is there. You will find the services-config.xml and the remotng-config.xml in the /railo-resin/webapps/ROOT/WEB-INF/flex directory. As with the previous version you will have to start the server at least once so that Railo creates the WEB-INF directory.

So after starting the server you can verify again that the MessageBroker Servlet is running by requesting <http://localhost:8600/flex2gateway>, if you receive the white page everything is good to go.

Using BlazeDS with the railo.war

If you have downloaded the railo.war only, you also have to add the MessageBroker Servlet like

we did with the Railo Express Server version. If you want to follow along, I downloaded and unzipped the railo-3.1.2.001.war into tomcats webapp directory. This is the same tomcate I used when explaining the basic setup of BlazeDS at the top of this page.

Start tomcat once, so that Railo will populate the WEB-INF directory. Now again all you have to do is to add the MessageBroker Servlet code, this time the easies one is to add it to the /tomcat/webapps/railo-3.1.2.001/WEB-INF/web.xml:

```
MessageBrokerServlet
flex.messaging.MessageBrokerServlet

    services.configuration.file
    /WEB-INF/flex/services-config.xml
```

1

```
MessageBrokerServlet
/flex2gateway/*
```

and restart tomcat. If you now request <http://localhost:8080/railo-3.1.2.001/flex2gateway> you should again have the white page.

Running multiple Railo Webs with BlazeDS

As we did in the last example, it is perfectly possible to run independent versions of BlazeDS for different websites. For example in the last example we had a blazesd and a railo-3.1.2.001 web running under the same tomcat.

But if you want to run multiple Railo webs on the same Railo server (with one global sever admin), you need to setup BlazeDS in every web separately and you also have to give it independand messagBrokerId's. BlazeDS will then manage instances of itself for your different Railo webs. To explain this we will add two new web apps to the Railo Resin Server example we used above.

Under /railo-resin/webapps/ add two directories called railo1 and railo2, copy the index.cfm files from the /railo-resin/webapps/ROOT directory into these two new apps and restart the server. Railo created two new webapps with independent web admins and one overall server admin for you. But if you look at the resin console output, you will recognise that the MessageBroker Servlet crashed with an unavailable exception and that it is already defined with messageBrokerId '__default__'.

On Resin calling <http://localhost:8600/railo2/flex2gateway> will already work, but for example on JBoss it is unreliable. So, to avoid the error you have to add web.xml files into both new railo1 and railo2 directories. Then add the MessageBrokerServlet to it using an independent messageBrokerId for every web. For example for railo1 in the /railo-resin/webapps/railo1/WEB-INF/web.xml add:

```
MessageBrokerServlet
flex.messaging.MessageBrokerServlet

    services.configuration.file
    /WEB-INF/flex/services-config.xml
```

```
messageBrokerId
railo1messageBroker
```

1

```
MessageBrokerServlet
/flex2gateway/*
```

You actually have to do this for any additional web context you add to a Railo multi web setup.

In the examples above, I always used the `/flex2gateway/*` mapping, this is because ColdFusion defines this as the default, but you can use whatever you want. Every Railo web gets its own set of flex config files which means you can setup different BlazeDS endpoints or even destinations for every web.