



Integral Information Solutions GmbH

Schickardstr 32 • D-71034 • Böblingen • Germany

## **Tutorial: Using the FusionReactor JDBC Driver Wrapper**

Doc. Rev. 242, 22 January 2008

---

## Trademarks and Warranties

FusionReactor, the FusionReactor logotype, and the Integral logotype are trademarks of Intergral Information Solutions GmbH and may not be used without permission. Other trademarks are the property of their respective owners.

The FusionReactor software product, including the FusionReactor JDBC Driver Wrapper is commercial software and may not be redistributed except with the express written agreement of Intergral Information Solutions GmbH. The software may only be used in accordance with the appropriate FusionReactor license agreement.

To the fullest extent applicable by law, Intergral Information Solutions hereby disclaims all warranties, including but not limited to the warranty of merchantability and the warranty of fitness for a particular purpose.

## Feedback

We welcome feedback on all our products and publications. Please e-mail them to [support@fusion-reactor.com](mailto:support@fusion-reactor.com) and we will address them as quickly as possible.

Published in Germany

Copyright © 2005-2008 Intergral Information Solutions GmbH

All Rights Reserved

---

FR 3.0.0

## Table of Contents

<b>Introduction</b> .....	<b>4</b>
User Guide.....	4
<b>Basic Setup</b> .....	<b>5</b>
Setting up a wrapped datasource.....	5
<b>Gathering metrics from ColdFusion pages</b> .....	<b>8</b>
Why did the query run in 0ms?.....	9
<b>Pathologically bad pages</b> .....	<b>10</b>
Using the Row Limiter.....	11

## Introduction

This document will take you through the steps install the FusionReactor JDBC Driver Wrapper, and provide you with a few examples of how it can be used.

The intention of the Wrapper is to provide a thin layer between J2EE applications and JDBC-accessible databases, in order to intercept and observe the interaction between them.

The Wrapper can then report various useful metrics to FusionReactor, and can also step in to prevent runaway queries. The Wrapper is fully integrated into the FusionReactor Administrator, and you can easily see all the queries which ran during a request, together with their runtime metrics and row counts.

Although the Wrapper is equally useful for JSP and Servlet deployment, this tutorial will illustrate the usage of the Wrapper under ColdFusion.

## User Guide

The Wrapper has its own user guide, which comprehensively details all possible options, together with the JDBC URL syntax and error messages.

You should consult this User Guide when working through the tutorial. It's installed with FusionReactor, and you can find it (under Windows) in your Start menu, in the FusionReactor program group. The JDBC Driver Wrapper User Guide is the definitive reference for the driver.

## Basic Setup

### Setting up a wrapped datasource

The first step in using the Wrapper is to find your ColdFusion datasource. This datasource should be working prior to applying the Wrapper. Find your datasource within the ColdFusion Administrator's 'Data Sources' section. Here's ours:

The screenshot shows the configuration page for a Microsoft SQL Server datasource named 'frtest'. The fields are as follows:

Data & Services > Datasources > Microsoft SQL Server	
Microsoft SQL Server : frtest	
CF Data Source Name	frtest
Database	frtest
Server	int0006
Port	1433
Username	sa
Password	***** (16-character limit)
Description	
<input type="button" value="Show Advanced Settings"/> <input type="button" value="Submit"/> <input type="button" value="Cancel"/>	

You can see that the example will use a built-in Macromedia driver to connect to the 'frtest' SQL Server database on int0006.

In order to have FusionReactor gather metrics for this connection, we must 'wrap' the driver. This will involve converting it to an 'other' type driver.

In order to wrap the driver, you'll need to obtain the **JDBC URL** for your existing datasource. The easiest way to do this is to click on **Server Settings -> Settings Summary**, then scroll down until you find the **existing** datasource. Copy the **JDBC URL**.

frtest	
CF data source name	frtest
Description	
Driver	MSSQLServer
JDBC URL	jdbc:macromedia:sqlserver://int0006:1433;databaseName=f
Username	sa
Login timeout	30 seconds
Long text buffer size	64000
Timeout	1200 seconds
Maintain connections	Yes
Interval	420 seconds
Restricted SQL operations	
Disable connections	No

In ColdFusion Administrator, add a new driver called 'frtest-wrapper', of type 'other'. In the **JDBC URL** field, you will enter the JDBC URL you copied from the Settings Summary page earlier, wrapped using FusionReactor's JDBC Wrapper syntax (more details below). You can refer to the FusionReactor JDBC Driver Wrapper User Guide to get some examples of JDBC URLs for the Macromedia builtin drivers.

Here's the form with our previous connection 'wrapped' with the FusionReactor JDBC Driver Wrapper.

The important things to note here are:

1. Since we're using an 'other' type driver, we used the JDBC URL we copied out of the Settings Summary page.

```
jdbc:macromedia:sqlserver://int0006:1433;databaseName=frtest;SelectMethod=direct;sendStringParametersAsUnicode=false;MaxPooledStatements=1000
```

2. We've added the FusionReactor JDBC Driver Wrapper text to the JDBC URL:

```
jdbc:fusionreactor:wrapper:{ jdbc:macromeida:... URL }
```

3. We've add the FusionReactor Driver Wrapper class name:

```
com.integral.fusionreactor.jdbc.Wrapper
```

4. The username and password are unchanged.

Since we're using a builtin Macromedia driver, there's no need to supply the 'driver' argument to the FusionReactor JDBC URL – CF will pre-load this driver for us. If we were using a third party driver, we would simply add the classname to the JDBC URL, like this:

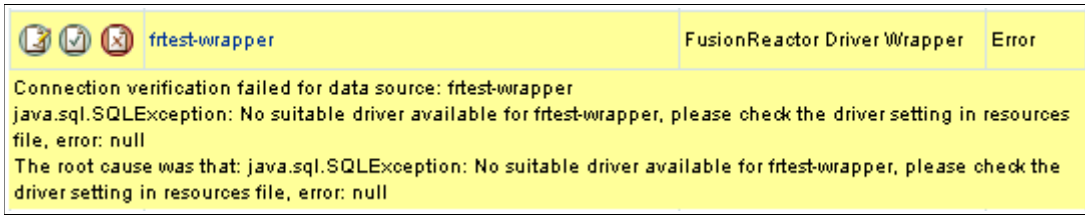
```
jdbc:fusionreactor:wrapper:{ jdbc:mysql:... URL };  
driver=com.somecompany.Driver
```

When you submit this form, ColdFusion will load our driver and make sure it can access the wrapped database. If all goes well, you'll see an 'OK' in the status column:

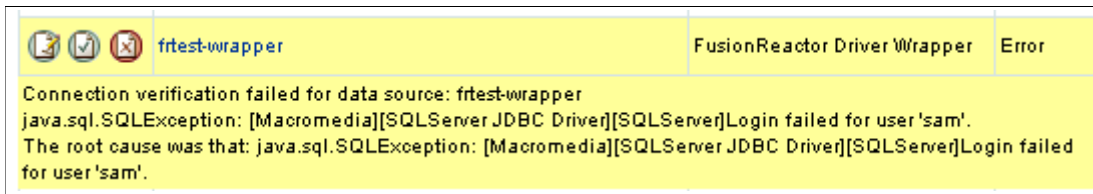
	frtest-wrapper	FusionReactor Driver Wrapper	OK
--	----------------	------------------------------	----

If something goes wrong, you'll get plenty of information to help you fix it. If FusionReactor can't access the wrapped driver, it will give you a helpful message indicating where the problem lies. If the wrapped driver has problems accessing the database, FusionReactor will relay its message

so you can fix the problem. Here's an example of a problem – we got the name of the FusionReactor driver class wrong:



You can see that ColdFusion couldn't load a suitable driver. Here's an example showing an incorrect database username:



Simply click on the datasource name to go back to the form and fix the problems. Once the datasource is ready, you can begin using it to gather metrics.

## Gathering metrics from ColdFusion pages

Once you've set up the `frtest-wrapper` datasource and verified it, we can begin to explore the metric-gathering abilities of FusionReactor.

Using our 'frtest' database, we can easily write a ColdFusion query using our new datasource:

```
<cfquery datasource="frtest-wrapper" name="getusers">
    select * from usr
</cfquery>
<cfoutput>Read #getusers.recordCount# users</cfoutput>
```

As you can see, this code simply reads all the rows from our 'users' table, a fairly small dataset. Running this page (which I've called 'tutorial1.cfm') gives the following output:

```
Read 2595 users
```

If we go to FusionReactor's 'Request History' screen, we can see this request as the most recent.

	15:06:02.803 200 02-May-2006 127.0.0.1	44 jrpp-5	<a href="http://localhost:qa/tutorial/tutorial1.cfm">http://localhost:qa/tutorial/tutorial1.cfm</a>	312 Cur:(3%)16,284 0 Free:503,971
---	---	--------------	---	--------------------------------------

Clicking on the blue 'data' icon at the very left of the request brings up the 'Request Details' page. When that page has displayed, click on the 'JDBC' tab heading to bring up the JDBC Details page, which looks something like this:

Statement	Query Time	Total Time	Exec Time	Row Count								
1 [[select * from usr]]	15:06:02.506	297	0	2595								
<table border="1"> <thead> <tr> <th>Total Statements</th> <th>Total Time</th> <th>Total Exec Time</th> <th>Total Row Count</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>297</td> <td>0</td> <td>2595</td> </tr> </tbody> </table>					Total Statements	Total Time	Total Exec Time	Total Row Count	1	297	0	2595
Total Statements	Total Time	Total Exec Time	Total Row Count									
1	297	0	2595									

This page tells us the query text that ran (if you were using `CFQUERYPARAM`, the actual, real values would be displayed in the statement), the exact time the statement ran, the total time (from running the query to finishing with the data), the exec time (the actual time the database took to run the query – in this case 0ms) and the number of rows read out by your query.

The final block contains the totals for all the queries which ran on the page.

## Why did the query run in 0ms?

You may occasionally see Total Time and Exec Time values which are 0ms. There are a couple of explanations for this.

1. The query never actually ran against the database.

If the query has been run before, and if there were no conditions which invalidated the underlying driver's cache, the result set may be delivered without actually running the query on the database. This can save a lot of time, and yields an Exec Time of 0ms.

2. The query ran, but ran very quickly.

In some operating systems, notably the Windows Server line (including Windows XP) the highest resolution of a timer available to Java is 10ms. If the query ran faster than 5ms, the timer will return a rounded-down value of 0ms. This is unfortunately unavoidable.

If you have more queries which run on a page, they'll be listed in the order they complete.

## Pathologically bad pages

Let's have a quick look at a couple of pathologically bad queries – designed to produce lots of data. Here's our test query:

```
<cfquery datasource="frtest-wrapper" name="getusers">
    select * from usr u, obj o
</cfquery>
<cfoutput>Read #getusers.recordCount# users</cfoutput>
```

The engineer is trying to work out which user (u) owns which object (o) but has gone for his lunch and on his return, forgets to add a constraining 'where' clause.

This query is going to select a cross join (cartesian product) of all Users against all Objects. The user table, as we've already seen, contains 2595 rows, and the object table contains 24799 rows. This will yield a Cartesian product of  $2595 \times 24799 = 64,353,405$  rows.

Oblivious to the error, we run the page and notice it's taking a long time to complete. In the FusionReactor Administrator, we can find the currently-running request:

		15:24:14.987 02-May-2006	127.0.0.1	47 jrpp-6	<a href="http://localhost/qa/tutorial/tutorial1.cfm">http://localhost/qa/tutorial/tutorial1.cfm</a>	40,327	Cur:(2%)12,492 Free:507,763
---	---	-----------------------------	-----------	--------------	---	--------	--------------------------------

... and click on the blue 'data' icon (top right of the icon block) to bring up the 'Request Details' page for this request. Once that page has appeared, we can click on the JDBC tab to bring up the JDBC Details page, which looks like this:

Currently Running Statement	Query Date	Query Time
[[select * from usr u, obj o]]	15:24:15.18	138734

It's now clear that our statement is causing the problem – since it's listed as the 'Currently Running Statement'. We can then use the red 'X' icon on the Running Requests page to kill this request. But what if we wanted to limit the number of rows retrieved to some 'hard' value, to prevent runaway queries in production?

## Using the Row Limiter

The FusionReactor JDBC Driver Wrapper has a builtin Row Limiter, which is configured on the Data Sources page of the CF Administrator.

In the CF Administrator, edit the 'frtest-wrapper' data source, and append

```
;rowLimit=10000
```

to the JDBC URL, making it:

```
jdbc:fusionreactor:wrapper:{jdbc:macromedia:sqlserver://int0006:1433;databaseName=frtest;SelectMethod=direct;sendStringParametersAsUnicode=false;MaxPooledStatements=1000};rowLimit=10000
```

Submitting that form will make sure the syntax is correct. The Driver Wrapper will now limit the number of returned rows to 10,000.

If we run our rogue page again, we can see that it only reads 10,000 rows – at which point the Row Limiter activates to prevent the query running away:

Read 10000 users

... and the JDBC Details page for the request also reflects the limit:

Statement	Query Time	Total Exec Time	Row Count
1 [[select * from usr u, obj o]]	15:36:39.403	3391	31 10000