Adabas Client for Java

Frequently Asked Questions

Version 2.0

November 2017

ADABAS & NATURAL

This document applies to Adabas Client for Java Version 2.0 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: ACJ-FAQ-20-20171103

Table of Contents

1 Frequently Asked Questions

Adabas REST Interface

Why do I get the message "ERROR: Forbidden"?

The message "ERROR: Forbidden" might be displayed when you try to access a local database. If you receive this message, you should edit the file *config.xml* in the directory *...\AdabasRestInterface\configuration* as shown below in order to allow your browser to access your database.

```
<!-- DatabaseAccess tag defines the databases the server is allowed to attach too.
 -->
 <DatabaseAccess>
  <Database dbid="nnn"  />
 </DatabaseAccess>
```

Enter the DBID of the database that you want to access in the line `<Database dbid="nnn" />`, then save the file and restart the Adabas REST Interface.

Note: Depending on your browser settings, you may have to stop and restart your browser. Otherwise, it may not be possible to set the required user credentials again.

How can I change the default administrator account for the Adabas REST Interface?

If you want to change the default administrator account, you should edit the file *realm.properties* in the directory *...\AdabasRestInterface\configuration* as shown below.

Modify the line

```
Administrator: MD5:70682896e24287b0476eff2a14c148f0, sagadmin
```

replacing the string `Adminstrator` with the name of the account that you want to use. Then save the file and restart the Adabas REST Interface.

How can I add new accounts for Adabas REST Interface authentication?

Intructions about how to do this are contained as comments in the header of the file *realm.properties* in the directory *...\AdabasRestInterface\configuration*.

Adabas Data Designer

Why is the Finish button greyed out (inactive) when I create a map definition using the mis.fdt example file?

The file *mis.fdt* in the directory *demodb* does not contain any long names for fields, and consequently the button **Finish** is not activated.

Attempting to execute one of the query mapping java example files returns the message "EmployeeMap does not exist or is not unique" - what other restrictions are known?

The prerequisites for using the Adabas Data Designer are described in the section *Adabas Client for Java - Concepts and Use Cases*: the database should be running and a metadata file already generated. Execute *GenerateExampleMappings* in Eclipse, and check for the following messages:

```
sample EmployeeMap for Adabas demo file 11 defined
sample VehicleMap for Adabas demo file 12 defined.
```

Now try to execute the query mapping java example file again.

Error Messages and Codes

Where can I find information about error messages and codes?

A list of ACJ messages and codes is included in the `QueryExceptionInfo` class. For further information, please refer to the javadoc package provided with the product.

What causes the message "log4j:WARN No appenders could be found for logger."

Sometimes three warnings can be observed on the console:

```
log4j:WARN No appenders could be found for logger.
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.
```

They are caused by a missing "log4j.properties" configuration file. You must ensure that this file is a part of the class path.

Logging

How do I start logging ?

Any errors that might occur are logged by default. If the log4j logging facility is used (which is the default in Adabas Client for Java), a log4j.log file will be written.In order t o increase the log granularity, you can define the logging category in the log4j.properties file, e.g. to "DEBUG". There are several example configuration files provided with the product that define the logging category as "INFO".

How do I log a message and/or an exception?

As mentioned in the Release Notes for this version, the logging mechanism is based on the Simple Logging Facade for Java framework (for further information, please refer to http://www.slf4j.org/manual.html).

... SLF4J does not rely on any special class loader machinery. In fact, each SLF4J binding is hardwired at compile time to use one and only one specific logging framework. For example, the slf4j-log4j12-1.7.12.jar binding is bound at compile time to use log4j. In your code, in addition to slf4j-api-1.7.12.jar, you simply drop one and only one binding of your choice onto the appropriate class path location. Do not place more than one binding on your class path. ...

All necessary jar files are part of the lib folder provided to use log4j (for further information, please refer to **http://wiki.apache.org/logging/log4j1**) as underlying logging framework. An example property file *log4j.properties* is contained in the "classic" examples folder.

How do I activate logging in one of the examples provided?

Import the slf4j Logger and LoggerFactory in your application :

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

Create a final static logger and define a name for the logger :

```
final static Logger logger = LoggerFactory.getLogger(<your-classname>.class);
```

Then log your messages with different priorities, for example INFO, WARN, ERROR :

```
...

logger.info("Filenumber given : {} ", fileNumber);

...

catch (QueryException e) {
   logger.error("Catch Exception - reason : {}", e.getMessage());

...
```

In order to set logging level INFO and output to FILE, adapt the log4j.properties file accordingly :

```
log4j.rootCategory=INFO, LOGFILE

# File
log4j.appender.LOGFILE=org.apache.log4j.RollingFileAppender
log4j.appender.LOGFILE.File=log4j.log
```