Adabas Client for Java

Adabas REST Interface

Version 2.0.1

October 2018

ADABAS & NATURAL

This document applies to Adabas Client for Java Version 2.0.1 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-REST-201-20181015

Table of Contents

Preface v
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
2 Adabas REST Interface 5
Adabas REST Server Configuration 6
Adabas REST Server API 10

Preface

This documentation contains information about the Adabas REST server, together with details of how to use it.

The following topic covered:

Adabas REST Server	Explains the main concepts behind Adabas REST server and describes how to use it.

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Software AG Documentation Website

You can find documentation on the Software AG Documentation website at **http://documentation.softwareag.com**. The site requires credentials for Software AG's Product Support site Empower. If you do not have Empower credentials, you must use the TECHcommunity website.

Software AG Empower Product Support Website

If you do not yet have an account for Empower, send an email to empower@softwareag.com with your name, company, and company email address and request an account.

Once you have an account, you can open Support Incidents online via the eService section of Empower at **https://empower.softwareag.com/**.

You can find product information on the Software AG Empower Product Support website at **https://empower.softwareag.com**.

To submit feature/enhancement requests, get information about product availability, and download products, go to **Products**.

To get information about fixes and to read early warnings, technical papers, and knowledge base articles, go to the **Knowledge Center**.

If you have any questions, you can find a local or toll-free number for your country in our Global Support Contact Directory at **https://empower.softwareag.com/public_directory.asp** and give us a call.

Software AG TECHcommunity

You can find documentation and other technical information on the Software AG TECHcommunity website at **http://techcommunity.softwareag.com**. You can:

- Access product documentation, if you have TECHcommunity credentials. If you do not, you will need to register and specify "Documentation" as an area of interest.
- Access articles, code samples, demos, and tutorials.
- Use the online discussion forums, moderated by Software AG professionals, to ask questions, discuss best practices, and learn how other customers are using Software AG technology.
- Link to external websites that discuss open standards and web technology.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 Adabas REST Interface

- Adabas REST Server Configuration 6
- Adabas REST Server API 10

The Adabas REST server included in the Adabas Client for Java package can be used to access Adabas data using any programming language that supports the HTTP protocol. The REST server provides an HTTP entry point to read (GET), create (POST), update (PUT) and delete (DELETE) Adabas records. The read (GET) functionality is set up in advance to be used with query search and sort parameters. The server can deliver the response in two formats, JSON and XML.

Adabas REST Server Configuration

This section provides information about how to configure the Adabas REST server before it is used for the first time.

- Adabas REST Server Prerequisites
- Adabas REST Server Configuration
- REST Server Startup
- REST Server Authentication

Adabas REST Server Prerequisites

In order to use the Adabas REST server to access a local Adabas database, the Adabas environment must be set up before the REST server is started for the first. If you want to access remote Adabas databases via Entire Net-Work, the corresponding Software AG Directory Server must also be configured.

Adabas REST Server Configuration

The Adabas REST server is provided with the default port set to 8190 for HTTP access and 8191 for HTTPS access. All configuration parameters are defined in the file *config.xml*, which is located in the configuration subdirectory of the Adabas REST Interface component. A default configuration file *config.default.xml* is provided, and can be renamed to *config.xml* for a first start of the server. The configuration file has the following content:

```
<RestServer>
    <Server>
        <Content directory="examples" />
        <Service port="8190" type="http" />
        <Service port="8191" type="https">
            <KeyStore file="keys/keystore.jks" />
            <KeyPassword password="test123" />
        </Service>
        <LoginService class="" type="hash" />
        <Shutdown passCode="shut123" />
    </Server>
    <Directory url="file:xtsurl.cfg" />
    <Mapping>
        <Database dbid="24" file="4" />
```

```
        <Database dbid="23" file="250" />
    </Mapping>
    <DatabaseAccess global="false">
        <Database dbid="1234" />
    </DatabaseAccess>
</RestServer>
```

Server

The Server element defines the basic attributes for running the REST server.

Content directory
This defines the directory of static HTML files which are used to provide a RESTful service. The examples subdirectory of the Adabas Rest Interface provides an example application to help you learn more about accessing Adabas using the REST interface.

Service port
This sets the TCP/IP listen port for HTTP and HTTPS requests. The access URL for the example application has the following form: https://localhost:8191/index.html. For HTTPS access, SSL certificates are necessary. The `KeyStore` and `KeyPassword` attributes are used to define the necessary settings.

LoginService
This defines the authentication method used by the REST server. The possible values for `type` are:

- hash: the passwords are encoded via MD5 hash.
- saf: the mainframe authentication ADASAF/RACF is used.

The `class` attribute can be used to specify an individual authentication class. If it is left empty, the built-in login service is used, that will provide the hash and saf authentication methods. See below for a description of how to write an individual authentication class.

Shutdown passCode
This defines the password that is used for the shutdown command for the REST server. There is a start menu entry on Windows to shutdown the server, and on Unix/Linux there is the shell script *stopAdaRest.sh*.

Note: the *keystore.jks* provided is only an example keystore with self-signed certificates. You must not use them in a production environment.

Directory

The *directory url* sets the path to the Software AG Directory Server (see Entire Net-Work). This might be a file reference *file:xtsurl.cfg* to a file that contains the access URLs of remote databases, or the URL of the Software AG Directory Server (e.g. tcpip://<host>:4952).

Mapping

Mapping definitions (for long field names, see *Adabas Data Designer* for more information) are stored in an Adabas file. Each database can hold one mapping file, but more than one database can have a mapping file. The Database tag is used to make these files known to the RESTful service.

DatabaseAccess

The `DatabaseAccess` attribute determines whether direct database references are allowed or not. If DatabaseAccess is set to false, it is not possible to use the http://<rest server>:<port>/rest/db for direct access to the database, and each database needs to be explicitly enabled for access. Two attributes are used to set the dbid:

- `dbid` is used to set the database ID;
- `url` is used to set an access path to a remote database (e.g. for a database 2000 on the mainframe url="2000(tcpip://1.2.3.4:3000)")

REST Server Startup

The REST server is started with the script *startAdaRest.sh* on Unix/Linux platforms, or with *startAdaRest.bat* on Windows platforms. On Windows, there is a Start Menu Entry for starting and stopping the REST server. Starting the server will open a console window to show the REST server output. Additional logging will be peformed in the temp subdirectory. After a successful start, the *server.console log* should look something like this (example):

```
    [Setting environment for Adabas Client for Java]
    C:\SoftwareAG\AdabasClientForJava\INSTALL\..\AdabasDataDesigner
    [done]
    JAVA_HOME: C:\SoftwareAG\jvm\jvm
    SERVER_HOME: C:\SoftwareAG\AdabasClientForJava\AdabasRestInterface

    "C:\SoftwareAG\jvm\jvm\bin\java" -cp ↩
"C:\SoftwareAG\AdabasClientForJava\AdabasRestInterface\lib\*" -Dfile.encoding=UTF-8 ↩
-Dserver.config=configuration/config.xml ↩
-Djava.io.tmpdir="C:\Users\bal\AppData\Local\Temp" -Xmx512m -Xms512m ↩
-DAdabasClientforJavaPidFile=C:\SoftwareAG\AdabasClientForJava\AdabasRestInterface\temp\server.pid ↩
-Djava.awt.headless=true -Dmain.class=com.softwareag.adabas.rest.RestServer ↩
-Dfile.encoding=utf-8 -Dcom.sun.management.jmxremote.port=8192 ↩
-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.authenticate=false ↩
-Dcom.sun.management.jmxremote.ssl=false com.softwareag.adabas.rest.RestServer
```

```
Feb 3, 2016 09:44:20 CET Starting Adabas REST server v1.1.0.0.1444
Map references:
[200](tcpip://10.20.114.24:0)_1000

Use Adabas Directory Server : file:xtsurl.cfg
Start HTTP server on port 8190
Start HTTPS/SSL server on port 8191
HASH login used, realm file is configuration/realm.properties
```

The REST server example page can be accessed via http://localhost:8190 or https://localhost:8191. Log in with user *Administrator* and the password *manage*.

REST Server Authentication

The REST server supports a minimal authentication realm. In order to set up the authentication using MD5 hash encoding, modify the *realm.properties* from the configuration subdirectory as required. By default, the user *Administrator* with password *manage* is defined. The *realm.properties* file defines users belonging to a role. The roles supported are *sagadmin* and *saguser*:

```
Administrator: MD5:70682896e24287b0476eff2a14c148f0, sagadmin

sag:  MD5:20384856e54267b7488eefea1ac1a8fa, saguser

user: MD5:d47f18dc7780fe47c24759714e2cd58f, saguser
```

The sagadmin role enables users to read, update, add and delete records in the databases, whereas the saguser role only allows records to be read.

If the *LoginService* hash was configured and the realm.properties file is removed or renamed, the REST server will not start. The *realm.properties* authentication is not used if ADASAF/RACF based authentication is configured.

Creating an MD5 Hash

This is only an example how to create the hash code, there are many tools available; this example uses the md5sum tool on Linux:

```
echo -n ThisIsMyPassword | md5sum
d47f18dc7780fe47c24759714e2cd58f -
```

Copy the hash code and paste it into the *realm.properties* file.

Writing an individual Authentication Class

The authentication class has to follow the principles of the jetty security (LoginService). Please see the **jetty documentation** for further details. The class has to implement the following interface:

```
/*
 * Copyright (c) 2015-2016 Software AG, Darmstadt, Germany and/or Software AG USA
 * Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates
 * and/or their licensors.
 * Use, reproduction, transfer, publication or disclosure is prohibited except
 * as specifically provided for in your License Agreement with Software AG.
*/

import org.eclipse.jetty.security.ConstraintSecurityHandler;

import org.eclipse.jetty.security.LoginService;

public interface ISecurityHandler {
  public ConstraintSecurityHandler getSecurityHandler();

  public LoginService getLoginService();
}
```

Adabas REST Server API

This section describes the Adabas REST Server API; a sample application that uses the REST interface is described in the following section.

A standard request URL to read Adabas data from a database has the following form:

```
http(s)://<host name>:<port>/<format>/<type>/<location reference path>
```

The host name is the name or an alias name of the machine on which the REST server is running. You must ensure that this name can be resolved using the standard naming service mechanisms. The port number is specified in the REST server configuration file (see **Adabas REST Server Configuration**).

- Format
- Type
- Location Reference Path
- Examples

- Adabas RESTful Example Application

Format

A request path starts with the format specification. The following formats are available:

Format	Explanation
`/rest/`	The `/rest/` path is the main point from which to query data. The default format when you use REST is JSON. However, the return type can be changed by using the "Accept:" header entry.
`/xml/`	The `/xml/` path returns XML by default. No "Accept:" header is needed. If the "Accept:" header is sent, the "Accept:" header is used.
`/json/`	The `/json/` path returns JSON by default. No "Accept:" header is needed. If the "Accept:" header is sent, the "Accept:" header is used.

The HTTP-Accept header is used to determine the response format. For the `/rest/` format the header is mandatory, for `/xml/` and `/json/` it is optional optional. If the header is sent, it is used to specify the format.

Type

The second element of the request path is the type specification. It can be `/db/`, to access a database directly, or `/map/`, to access the database via a map definition. The type path element is followed by the location reference path.

Note: a map definition always contains a reference to the database ID and file number, so it is not necessary to specify them in a request.

Location Reference Path

The HTTP location reference path consists of several parts that define the resource to be accessed. Depending on the type, the reference path consists of various path elements:

Type	Path Element	Explanation
`/db/`	1. Database ID 2. File number 3. [ISN] 4. [Adabas field]	The database ID and file number can be specified. If you want to access a specific record and know the associated ISN, the ISN can be addressed directly.
`/map/`	1. Map name 2. [ISN] 3. [Map field name]	A predefined map can be accessed directly by using the map reference.

In addition to the location reference, a number of parameters can be used to modify the request that is sent:

Parameter	Explanation
fields	Defines a list of field requests for the resulting data.
start	Defines the offset or ISN from which to start reading.
limit	Limits the number of returned entries.
sorted_by	Provides one field that will be used to sort the resulting data. Currently, the field used with sorted_by must be an Adabas descriptor.
search	With the search parameter 1. a standard search phrase can be provided. For example "AE='SMITH'" or "LastName='SMITH'"; 2. a complex JSON query with additional information can be sent to the server.

These parameters are specified in the standard HTTP fashion - '?fileds=AC,AE&limit=10'.

Examples

A request to read ISN 1 from database ID 4 and file 11 has the following form:

```
http://localhost:8190/rest/db/4/11/1
```

This request will return the following data in JSON format (default):

```
{"NrRecords":1,"FileRecords":-1,"Records":[{"ISN":1,"AA":"50005800","AB":{"AC":"SIMONE","AD":"","AE":"ADAM"},"AF":"M","AG":"F","AH":712981,"A1":{"AI":["26 ↩
AVENUE RHIN ET ↩
DA"],"AJ":"JOIGNY","AK":"89300","AL":"F"},"A2":{"AM":"44864858","AN":"1033"},"AO":"VENT59","AP":"CHEF ↩
DE ↩
SERVICE","AQ":[{"AR":"EUR","AS":963,"AT":[138]}],"A3":{"AU":19,"AV":5},"AW":[{"AX":19990801,"AY":19990831}],"AZ":["FRE","ENG"]}]}

 ↩
```

The same request with an HTTP-Accept header 'Accept: application/xml' (or use format specification '/xml/') will result in the following output:

```
<?xml version="1.0" encoding="UTF-8"?><Response><Record ISN="1"><AA ↩
sn="AA">50005800</AA><Group sn="AB"><AC sn="AC">SIMONE</AC><AE sn="AE">ADAM</AE><AD ↩
sn="AD"></AD></Group><AF sn="AF">M</AF><AG sn="AG">F</AG><AH ↩
sn="AH">712981</AH><Group sn="A1"><Multiple sn="AI"><AI sn="AI">26 AVENUE RHIN ET ↩
DA</AI></Multiple><AJ sn="AJ">JOIGNY</AJ><AK sn="AK">89300</AK><AL ↩
sn="AL">F</AL></Group><Group sn="A2"><AN sn="AN">1033</AN><AM ↩
sn="AM">44864858</AM></Group><AO sn="AO">VENT59</AO><AP sn="AP">CHEF DE ↩
SERVICE</AP><Period sn="AQ"><Entry><AR sn="AR">EUR</AR><AS sn="AS">963</AS><Multiple ↩
sn="AT"><AT sn="AT">138</AT></Multiple></Entry></Period><Group sn="A3"><AU ↩
sn="AU">19</AU><AV sn="AV">5</AV></Group><Period sn="AW"><Entry><AX ↩
sn="AX">19990801</AX><AY sn="AY">19990831</AY></Entry></Period><Multiple sn="AZ"><AZ ↩
```

```
sn="AZ">FRE</AZ><AZ sn="AZ">ENG</AZ></Multiple></Record></Response>

 ↩
```

A request to read only 20 records and only the fields AC and AE from database ID 4 and file 11 has the following form:

```
http://localhost/:8190/rest/db/4/11?fields=AC,AE&limit=20
```

If there is a map definition with the name "MY-EMPLOYEES-MAP", created with the Adabas Data Designer for the example file 11 (EMPLOYEES-NAT) of database 4, a request to read ISN 1 using this map definition has the following form:

```
http://localhost:8190/rest/map/MY-EMPLOYEES-MAP/1
```

This request will return the following data in JSON format (default):


```
{"NRecords":1,"FileRecords":-1,"Records":[{"ISN":1,"personnel-id":"50005800","full-name":{"name":"ADAM","middle-name":"","first-name":"SIMONE"},"mar-stat":"M","sex":"F","birth":712981,"full-address":{"post-code":"89300","address-line":["26 ↩
AVENUE RHIN ET ↩
DA"],"country":"F","city":"JOIGNY"},"telephone":{"area-code":"1033","phone":"44864858"},"dept":"VENT59","job-title":"CHEF ↩
DE ↩
SERVICE","income":[{"curr-code":"EUR","bonus":[138],"salary":963}],"leave-date":{"leave-taken":5,"leave-due":19},"leave-booked":[{"leave-start":19990801,"leave-end":19990831}],"language":["FRE","ENG"]}]}
```


Note: The long names for the fields defined in the map are returned rather than the short names (see the first example above).

A request to read only 20 records and only the two fields `first-name` and `name` from the map has the following form:

```
http://localhost:8190/rest/map/MY-EMPLOYEES-MAP?fields=first-name,name&limit=20
```

This request will return the following data in JSON format (default):


```
[illegible]

 ↩
```

Adabas RESTful Example Application

The Adabas RESTful example is an application that shows you how to use the Adabas REST interface. It is located in the examples subdirectory of the Adabas REST Interface installation directory, and as a 'war' file in the war subdirectory.

Prerequisites

Before you start with the example, you must prepare a database with the necessary files and mapping definitions:

1. Create a standard Adabas demo database with the `crdemodb` command, for example, *`crdemodb 4`*
2. Load the backup file *example.bck* into the database, (for example, use the following command sequence: *`adadbm db=4 delete=4-202`*, *`BCK001=example.bck`*, *`adabck db=4 restore=4-202`*), do not renumber the files. The backup file is located in the data subdirectory of the REST Interface installation.
3. Create two mappings with the Data Designer: *`TestMapEmployee`* for file 11, and *`TestMapVehicle`* for file 12. The required mappings are shown in the tables below; it is necessary to create the mappings exactly as shown.
4. Configure the REST Server in order to gain access to the database files and the mapping files (refer to ***Adabas REST Server Configuration*** for further information).

TestMapEmployee

Adabas Field	Long Name
AA	ID
AC	FirstName
AE	LastName
AO	Department
AS	Salary
AT	Bonus

TestMapVehicles

Adabas Field	Long Name
AD	Vendor
AE	Model
AF	Color

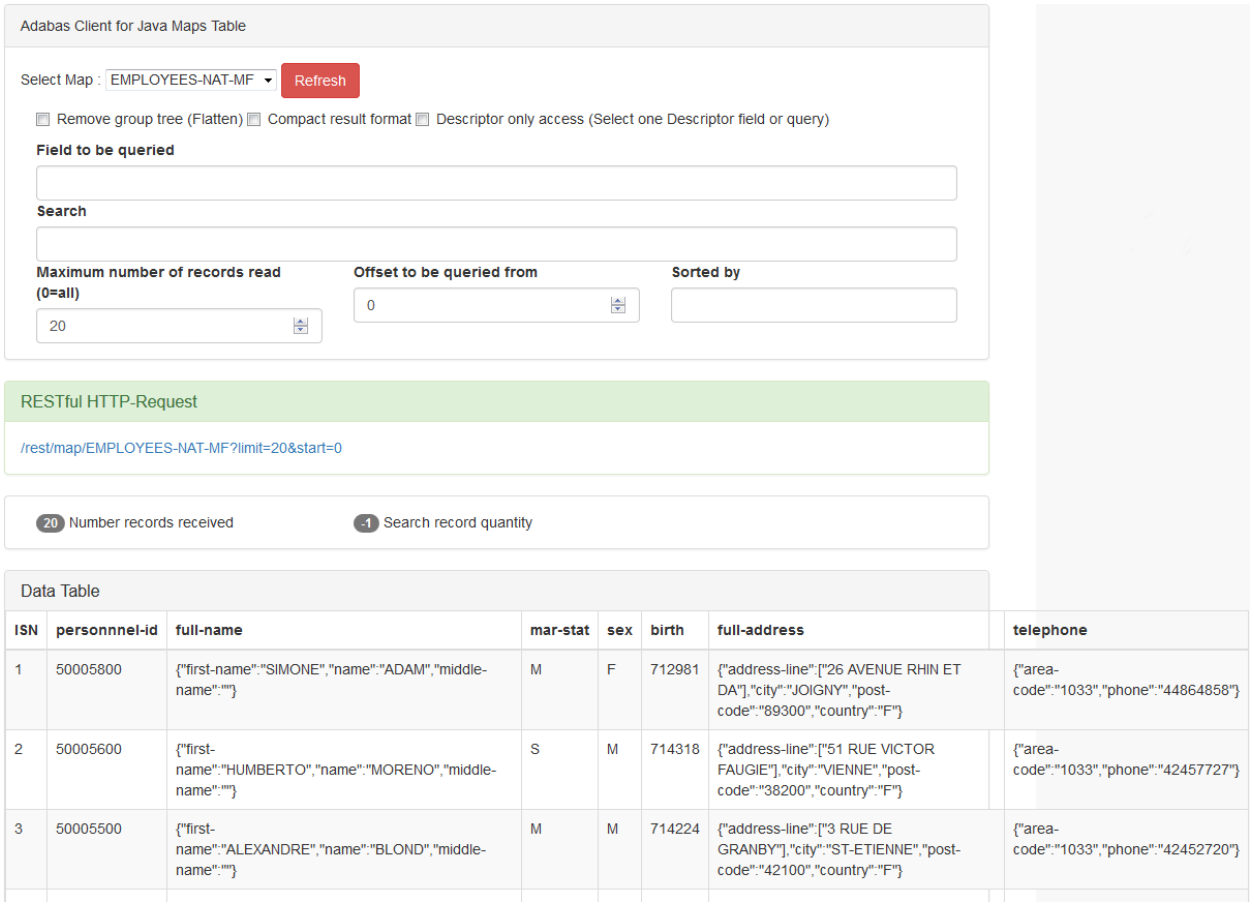
The Example

The first step is to choose the correct database. The example uses the DBID 24 as default. This can be changed by choosing the **Select demo database** button. If the DBID is to be changed persistently, the *index.html* file has to be changed accordingly (see AdabasRestInterface/examples directory). There are some predefined JSON queries that request data from the database and show the results in the lower part of the window.

Additionally, the URL of the query will be shown, and the result is displayed in JSON format. Click on the **Rest Interface URL:** or **Result:** button to see all of the data.

Choose the **Free Form** tab if you want to make more flexible requests, and the **Modify Record** tab for a demonstration of update and delete requests. Note that the data in the database will not be modified, it just constructs the URLs for PUT and DELETE and shows the JSON format for these type of requests.

Running the example will produce a screen similar to the one shown below:



ISN	personnel-id	full-name	mar-stat	sex	birth	full-address	telephone
1	50005800	{"first-name":"SIMONE","name":"ADAM","middle-name":""}	M	F	712981	{"address-line":["26 AVENUE RHIN ET DA"],"city":"JOIGNY","post-code":"89300","country":"F"}	{"area-code":"1033","phone":"44864858"}
2	50005600	{"first-name":"HUMBERTO","name":"MORENO","middle-name":""}	S	M	714318	{"address-line":["51 RUE VICTOR FAUGIE"],"city":"VIENNE","post-code":"38200","country":"F"}	{"area-code":"1033","phone":"42457727"}
3	50005500	{"first-name":"ALEXANDRE","name":"BLOND","middle-name":""}	M	M	714224	{"address-line":["3 RUE DE GRANBY"],"city":"ST-ETIENNE","post-code":"42100","country":"F"}	{"area-code":"1033","phone":"42452720"}