Adabas Client for Java

Adabas Client for Java - Concepts

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-INTRO-20-20171103

Table of Contents

Preface

This documentation describes the concept and components of the Adabas Client for Java. It is organized in the following sections:

Adabas Client for Java	General information on Adabas Client for Java
Adabas Data Designer	General information on Adabas Data Designer
Adabas Rest Interface	General information on Adabas Rest Interface

1 Adabas Client for Java

The Adabas Client for Java is an interface to Adabas for Java-based applications. The aim of the product is to make it easy for Java developers to write new business applications by using business-critical data stored in Adabas.

The idea behind the Adabas Client for Java is to provide a state-of-the-art interface to Adabas for applications either written in Java or using HTTP-based communications with the REST interface.

The Adabas Client for Java provides:

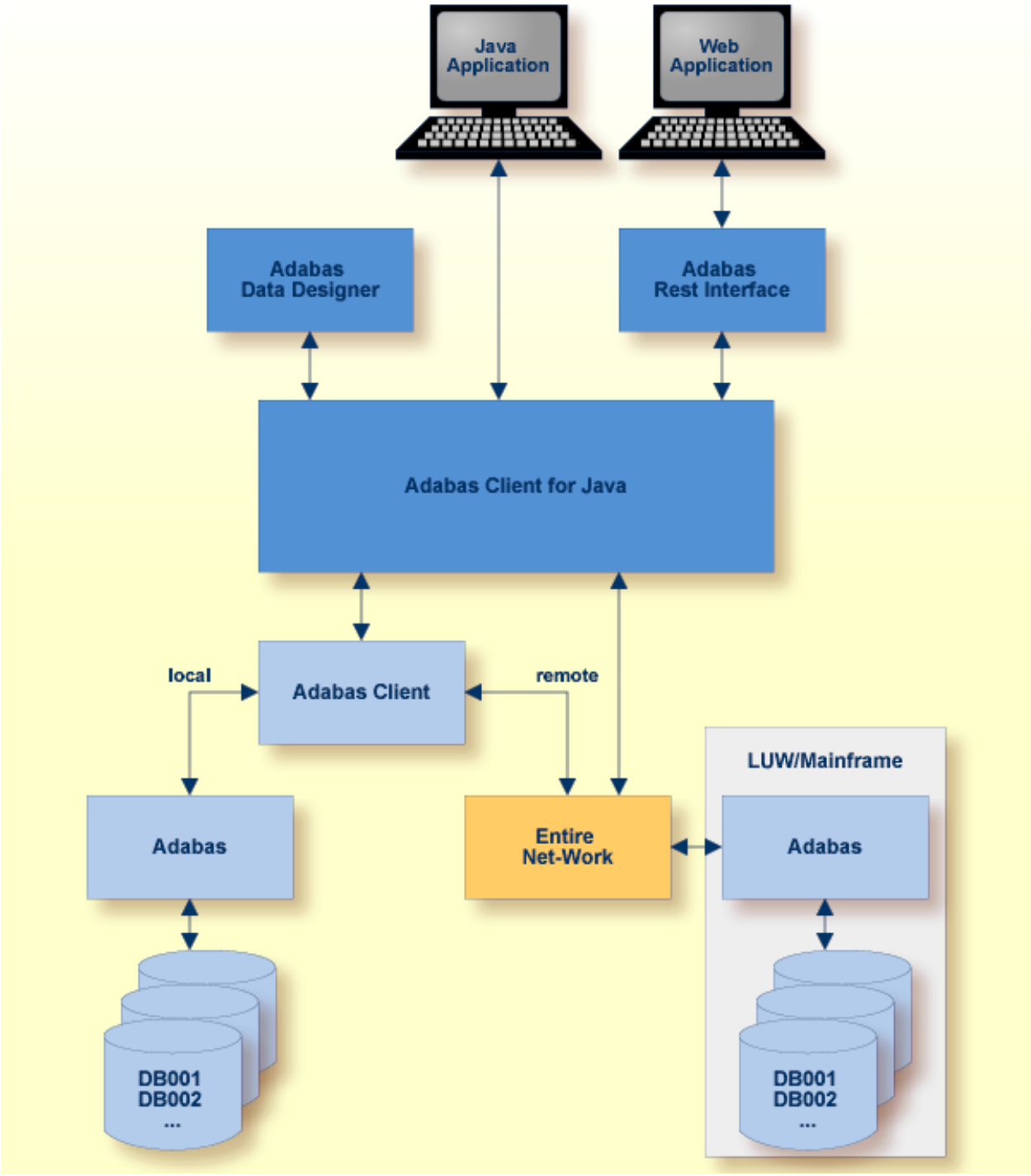
1. A high-performance API to access Adabas data in a Java application;
2. The ability to interface with new Software AG products, e.g. Terracotta;
3. State-of-the-art standard communication protocols;
4. The potential to integrate into other frameworks or products.

The product consists of three individual components:

1. The Java-based API itself;
2. The Adabas REST Interface
3. The Adabas Data Designer

Communication between the components is based on TCP/IP. It is, however, also possible to use Software AG's middleware package Entire Net-Work to communicate with Adabas; this is necessary if you want to access remote Adabas databases that are, for example, located on a mainframe.

The following picture shows the components of the Adabas Client for Java and how they are connected:


Java Application
Web Application
Adabas Data Designer
Adabas Rest Interface
Adabas Client for Java
local
Adabas Client
remote
LUW/Mainframe
Adabas
Entire Net-Work
Adabas
DB001
DB002
...
DB001
DB002
...

Java Development with the Adabas Client for Java API

The base for all of the Adabas Client for Java components (the Adabas REST Interface and the Adabas Data Designer) is an API for developing Java applications. Because the vocabulary of Adabas and the structures in Adabas are different to Java and other databases, there are classes available to encapsulate the appropriate functions and structures.

There are basic classes which handle the operations to and from Adabas. The main classes are `ReadRequest`, `StoreRequest` and `DeleteRequest`.

- `ReadRequest` handles all read operations, including queries and searches. It can be used to retrieve data from an Adabas database.
- `StoreRequest` handles all single store and update operations.
- `DeleteRequest` handles delete operations of records.

The `ReadRequest` and `StoreRequest` classes, that are required to read or modify data, implicitly close the session after each request that is sent. The Java application itself decides if the session needs to be ended, or whether to either backout or commit the transaction, or to close the database connection. In advance, the record could be put into hold status. For further details, see the JavaDoc for the `AdabasTarget` class.

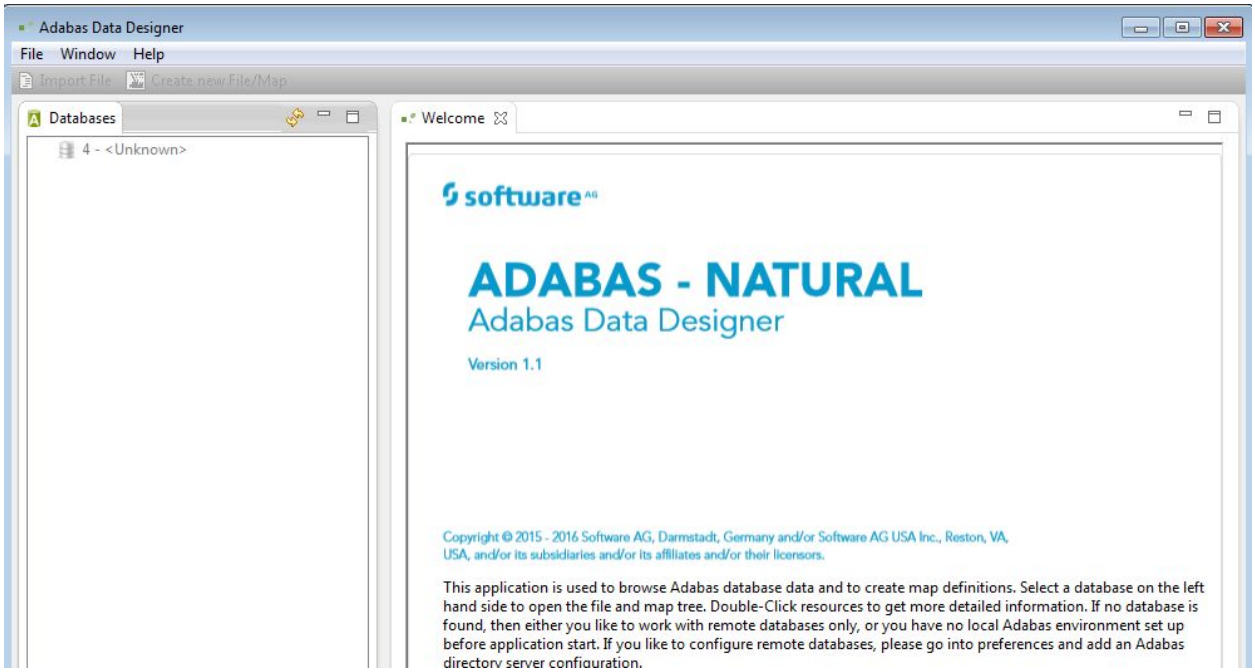
Within the basic classes, the fields (this is the Adabas equivalent to SQL columns) are defined using either a short name or a long name. The Adabas Client for Java API can be used to create a view (catalog or map) in which, analogous to SQL tables, a meaningful name (so-called long name) can be used for the column and for the view itself. In Adabas, the file name (similar to SQL tables) is limited to 16 characters, and the field name is specified by 2 characters (short name) - both of these restrictions can be lifted in the view definition by mapping the long names to the short names. The map definitions are stored in an additional file in the Adabas database.

2 Adabas Data Designer

The Adabas Data Designer is based on the Eclipse Rich Client Platform (RCP). The tool can read the FDT data of an Adabas file and create new map files. Currently, an XSD, FDT or DDM file is needed to generate a map . The XSD definition automatically creates a new Adabas file based on the XSD definition.

The Adabas Data Designer is used to configure the Java API maps, as well as to browse the data in Adabas files. The combination of configuring maps and parallel browsing of the data helps to provide an overall view of the combination of metadata and data.

The Adabas Data Designer can be used to import DDM defined views into the Adabas Client for Java definitions. DDMs exported with the Natural Object Handler (SYSOBJH) can be imported in the Adabas Client for Java Map definition. All Adabas Client for Java configuration and Map definitions are stored in an Adabas file in the Adabas database


Adabas Data Designer
File Window Help
Import File
Create new File/Map
Databases
4 - <Unknown>
Welcome
software AG
ADABAS - NATURAL
Adabas Data Designer
Version 1.1
Copyright © 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.
This application is used to browse Adabas database data and to create map definitions. Select a database on the left hand side to open the file and map tree. Double-Click resources to get more detailed information. If no database is found, then either you like to work with remote databases only, or you have no local Adabas environment set up before application start. If you like to configure remote databases, please go into preferences and add an Adabas directory server configuration.

3 Adabas REST Interface

The Adabas REST Interface included in the package can be used to access data from any programming language that supports HTTP requests. The Adabas REST Interface provides an HTTP entry point to read (`GET`), create (`POST`), update (`PUT`) and delete (`DELETE`) Adabas record data. 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 text formats, JSON and XML.

The data can be accessed in two ways. One way is to use a classic Adabas database reference with database ID, file number and field name. A second way is to use the Adabas Client for Java to view definitions, which in turn requires a map.

A standard request URL to read data has the following form:

```
http://<host name>:<port>/rest/<location reference>
```

A request to read JSON data out of database ID 24 and file 11 with ISN 1 would look like this:

```
http://localhost:8190/rest/db/24/11/1
```

This request will return JSON data that looks as follows:

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

If the HTTP header entry `"Accept: application/xml"` is sent, the result will look like this:

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

A set of filter, search and modification functions is provided with the package.