JDOOCS – A JAVA LIBRARY FOR DOOCS

K. Rehlich , DESY, Hamburg, Germany
V. Kocharyan, YerPhI, Yerevan, Armenia.

Abstract

To give a possibility to access data from different computer platforms we have developed a client part of DOOCS library in Java language (JDOOCS). It is fully identical to the DOOCS Client API and allows to read and write data from/to servers supporting DOOCS and TINE protocols. For testing of the JDOOCS library we have written a generic application in Java that allows to access all data in the control system.

1. INTRODUCTION

The DOOCS [1] is the control system of TTF (TESLA Test Facility), where every type of hardware device is presented as an object. It provides communication via a standard set of data and address objects transferred by Open Network Computing / Remote Procedure Calls (ONC/RPC) from Sun. It's available on Solaris, SunOS, LINUX and Windows.

The main goal to create Java libraries is to have a possibility to include more computer platforms in the control system and develop the Web technologies. The server-client data exchange of the DOOCS supports more then thirty data types, which were generated by the *jgen.jar* application, and is inserted into the *jdoocs.jar* library. ONC/RPC of the JDOOCS is based on the ACPLTea (Aachen University of Technology, Germany) library packages. The library and applications were written by means of SUN Forte JDK and the documentation was also generated. The applications are tested on Solaris, LINUX and Windows platforms.

2. THE LIBRARY STUCTURE

The Java JDOOCS library, just as the C++ DOOCS client library, contains three main classes *EqCall*, *EqAdr*, *EqData* and the extra frames, which collect the RPC request in a local database (*EqServices* and so on).

The *EqCall* class is the client interface to the RPC communication. This class library does the actual transfer of the data from/to the devices and supports DOOCS as well as TINE protocols. The class has three communication methods:

EqData get (*EqAdr* adr, *EqData* data);

EqData set (*EqAdr* adr, *EqData* data);

EqData names (*EqAdr* adr).

All calls return an *EqData* class object. The API has access to a name server (ENS), which resolves the necessary information about the servers. These data are stored inside the server table so that just one call to the name server is needed. After this information is stored in the library's local database. The hosts where the name servers run may be passed via ENSHOST environment variable.

The *EqAdr* class is the address definition. This class library is the address part of the communication and is composed of four fields of the address string. The class methods allow to initialize and manipulate the address string.

The *EqData* class is the interface to the data objects transferred from the java application to the device server and the returned data too. The class methods allow to send data in its native form and converted on the receiver side, since the EqData class contains data type and request time information. For instance, a get call specifies an address and a data object to be sent to the server and gets a result object from the server.

```
import ttf.doocs.clnt.*;

……..……..……..……..……..……..…….
// create the required objects
EqAdr    ea = new EqAdr();
EqData  ed = new EqData();
EqData  result  = null;
EqCall  eq = new EqCall();
// fill the address class
ea.adr("TTF.RF/ADC/HOR.ADC1/CH00.TD");

// init sending object

String data  = "-1 0 4 100";

ed.set_type(eq_rpc.DATA_IIII);

ed.set_from_string(data);

//do call

result = eq.get (ea, ed);
System.out.printfln(result.get_string() );
```

As a result we'll get from the server an error message or data presented in a string form.

3. RPC_UTIL - JAVA APLLICATION

As a test application, we have written a simple, but powerful java Swing tool to read or change all values in the DOOCS control system. The tool was archived in *rpc_util.jar* java archive. To run the application, type: *java –jar rpc_util.jar*. Using this tool is simple: by clicking on the desired "Facility", then "Device", "Location" and finally "Property" fields , you choose the channel and can see the requested value in the "Result" line. Then click on the "Read" button to update the chosen value. It also sets the received data type in the JComboBox. To send data to the server, choose the data type from JcomboBox, put data into the textfield first, and then click the "Send" button.

Figure 1: The Java tool to communicate with all DOOCS data

We have the *ttfplot* package for the visualization of the data array. The *ttfplot* package receives the data via RPC calls :

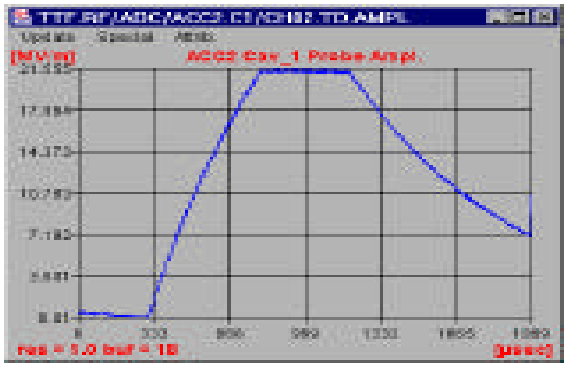

Figure2: The spectrum plot of an amplitude channel

or via request to a special servlet:

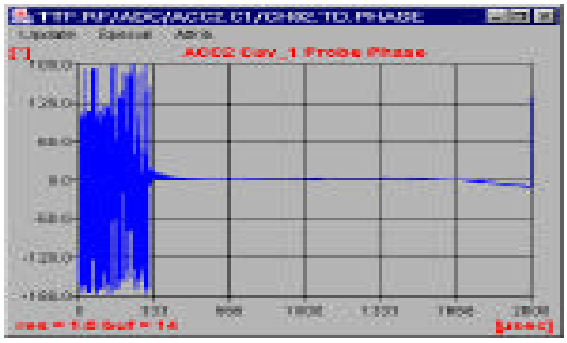

Figure3: The spectrum plot of a phase channel

We still didn't solve the authorization problem for set requests from the non-UNIX environment, but it's supposed to be solved via a special call to the name server.

The response time of the RPC calls made by this tool was also analyzed - it has a similar speed as the C client API calls.

4. JDOOCS IN WEB

As a Web implementation *jdoocs.jar* has been included in Tomcat libraries path on the TTF Logbooks web server. It allows to write servlets, which are performing RPC calls and transferring the requested data through the Web. The RPCTest applet is shown in the picture below, it displays data received from the servlet. This applet gives a possibility to view and control all the servers running in the DOOCS system remotely.

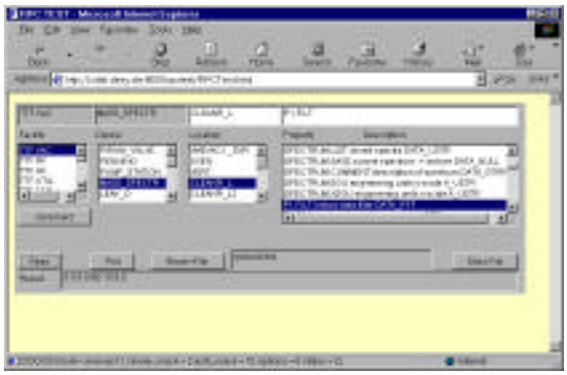

Figure 4: The RPCTest applet in the browser window

5. CONCLUSION

This library and the applications based on it allow to perform a full control of TTF status and in the future also TESLA control. The developing Java and web technologies will provide us with useful tools to display and control accelerator equipment on any computer platform.

We would like to express special thanks to P. Duval for supplying the *tine.jar* library which allowed to support the TINE protocol in JDOOCS.

6. REFERENCES

[1] K. Rehlich et.al., “DOOCS: an Object Oriented Control System as the Integrating Part of the TTF Linac”, Proceedings of ICALEPS97, Beijing China)