

Technical Overview of the ASL Database

Josiah Yoder Ruth Devlaeminck

June 2007

This document is also available on the web in HTML form at
<http://cobweb.ecn.purdue.edu/RVL/Database/ASL/overview/>

Contents

1	Introduction	1
2	The Server Components	2
2.1	HTTP Server	2
2.2	ASL Server	3
2.2.1	Compiling the ASL Server	3
2.2.2	Starting the ASL Server	4
2.3	MySQL Database Server	4
2.3.1	Restarting the mysql server	5
2.3.2	Use the "stable" executable locations	5
2.3.3	Socket file (/tmp/mysql.sock)	5
2.3.4	Username	5
2.3.5	Connection between MySQL and Java	6
2.3.6	Setting up the MySQL database	6
3	The Client Side	6
3.1	The Applet	7
3.1.1	Running the Applet	7
3.1.2	Compiling the Applet	8
3.1.3	Signing the Applet	8
4	Automatic Restart	9
5	Adding new users	11

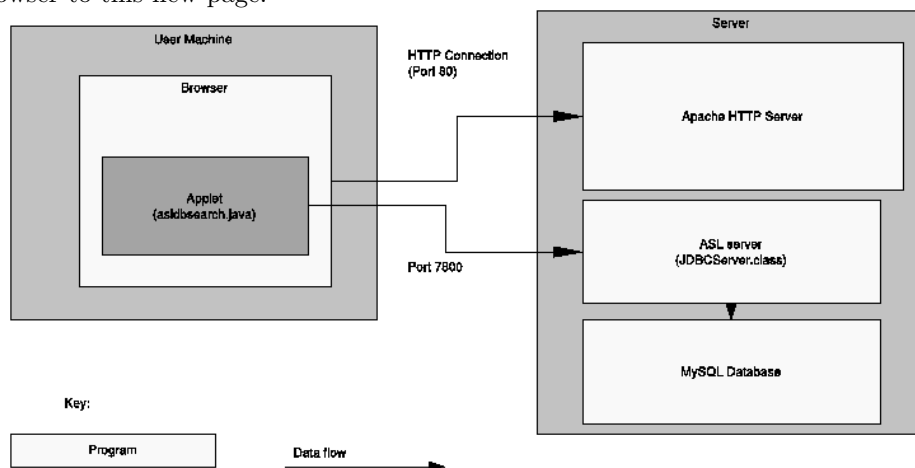
1 Introduction

The Purdue RVL-SLLL ASL Database is "an extensive database of American Sign Language (ASL) motions, handshapes, words and sentences." This doc-

ument describes the structure and workings of the web user-interface to the database.

Two additional documents provide important information on the data contained in the database. The technical report [1], lists each handshape and paragraph contained in the database and gives instructions on how to use the graphical user interface (GUI). The paper by Martinez et al [2] gives further details of how the images were obtained, including the lighting conditions and the background of the signers. Both documents are recommended to anyone working with the database.

The database is accessed primarily through a java applet which runs on the client side. This applet connects to a custom JDBC server (the "ASL Server"). This server performs queries to the database on request from the applet and creates a new webpage for the main query. The applet then directs the user's browser to this new page.



2 The Server Components

We discuss the server components first because it is necessary for them to be running before the client side can operate.

2.1 HTTP Server

This is a server running on cobweb.ecn.purdue.edu that is maintained by ECN. It serves files located on rv12.ecn.purdue.edu. It does not contain any active components, such as php scripts or Java Servlets. The HTTP server contains the main RVL website, a page with the database access applet, and all the pages with the ASL Server automatically generates.

2.2 ASL Server

Location data	
directory	/home/rvl2/d/asldata/ASLdatabaseSearch/server
Main file	JDBCServer.java
Additional files	ClientHandler.java Connect.java
JDBC connector	/home/rvl2/d/asldata/public-web/ mysql-connector-java-3.1.10-bin.jar
listening port	7800
log file	/tmp/JDBCServer.out

This server is a custom-built Java JDBC server. The server accepts connections from the applet using a simple protocol. It accesses the MySQL database through JDBC. It returns the intermediate results to the applet through the same protocol, but writes the result of the final query to a new web page. The name of the new page is then sent to the applet through the protocol.

The JDBCServer.java class creates a new ClientHandler for each request that is received, allowing multiple users to access the database at the same time. The Connect class is a small wrapper for accesses to MySQL through JDBC.

The following example illustrates the protocol used between the ASL Server and the applet client. The details of the protocol are best learned by reading the applet and server java source files, named asldbsearch.java and ClientHandler.java respectively.

```
Example interaction
Client: SELECT Subjects.Number FROM Subjects WHERE
        Subjects.Number = 14
Server: 14
Server: Done
Client: makeFile
Client: SELECT Main.Filename, Main.CompFile FROM Main WHERE
        Main.LightCond = 'D' AND ( Main.Paragraph = 1 )
        AND ( Main.Subject = 14 )
Server: 1637400138
Client: exit
```

In this example, the database returns the query results for the first query to the client. The second query is preceded by the command "makeFile", so the server creates the file and returns the ID used to create the file instead. The file's name is fileListNNNNNNN.html, where NNNNNNN is the number returned to the client. In this example, 1637400138 is the number returned, so the file's name is fileList1637400138.html.

These files are created in the directory /public-web/.

2.2.1 Compiling the ASL Server

All of the java files in this directory need to be compiled.

```
java *.java
```

2.2.2 Starting the ASL Server

To start the server, use

```
~/bin/jsrvr.chk
```

This script is included in Section 4.

The old-fashioned way to start the server, should it become necessary, is:

```
java -classpath ../home/rvl2/d/asldata/public-web/  
mysql-connector-java-3.1.10-bin.jar JDBCServer &
```

The server uses JDBC to access mysql, which is why the driver must be included using the jar file mysql-connector-java-3.1.10-bin.jar.

This command should be run as a single line with no space between the / and mysql. Generally,

```
~/bin/jsrvr.chk
```

should be used instead.

The server is automatically restarted. See Section 4 for details.

2.3 MySQL Database Server

This server is a standard MySQL database. In the ASL database, it accepts its connections from the local machine, not over the internet. The MySQL database video files are not stored directly in the database. Rather, the videos are stored in the file system. The mySQL database contains paths to the videos and descriptions of each video in a relational database format. Descriptions of the contents tables are available in [1].

Location data	
maintainer	Robot Vision Lab
machine	rvl2.ecn.purdue.edu
executable	/usr/local/libexec/mysqld
database files	/home/rvl2/a/mysql/
database	asldatabase
users	asldata
	readonly
tables	Main
	Glosses
	Paragrahs
	Subjects
log file	/tmp/mysqld.log

Users coming in from the web are made to be user 'readonly'.

The vidoes in the database are each named

xx-y-zz-u.avi

where the codes are given in the following table.

Video file naming conventions		
Signer code	xx	01-14
Type	y	M (for motion), H (for hand-shapes), A (for alphabet), N (for number), P (for paragraph)
File number	zz	1-43 (for motion), 1-10 (for paragraphs), 1-20 (for handshapes)
Lighting	u	D(iffuse), C(ontrast)
Examples		
01-M-01-C		Subject 1, Motion 1, Contrast lighting
01-A-D		Subject 1, Alphabet, Diffuse lighting

2.3.1 Restarting the mysql server

The server must be started with an option to direct it to the location of the database files.

The server is started with the command

```
/usr/local/libexec/mysqld -datadir=/home/rvl2/a/mysql/
```

The server is automatically restarted if it fails. See Section 4.

2.3.2 Use the "stable" executable locations

- /usr/local/libexec/mysqld - I use this program, it is the main server.
- /usr/local/bin/mysqld_safe - This is an alternative. It is a script that allows the mysqld to be automatically restarted if it crashes.

2.3.3 Socket file (/tmp/mysql.sock)

- It is sometimes possible to use a socket file instead of an external IP-port (e.g. /tmp/mysql.sock)
- We cannot create socket files because, as "The definitive Guide to MySQL5.0" clearly states: (p.726) "(An exception is Java clients, which do not support socket files.)"
- Using the standard port always opens up the server to vulnerabilities in mysql, even if password protection is enabled.
- Programs on the machine would need to be able to access the server through the socket file. This could be difficult.

2.3.4 Username

The mysql server should use the username that the database files are stored under.

2.3.5 Connection between MySQL and Java

One must download the `mysql-connector-java-3.1.10-bin.jar` files for the jdbc driver. This driver allows a jdbc program to access mysql. There is a copy of this jar file in the public-web directory, but I think we put another copy someplace else, don't remember where. Have to add these jar files to the java classpath, as described in the section "ASL server"

2.3.6 Setting up the MySQL database

Note: This section describes how to create the database. See 2.3 for the current location of the database.

To set up the server, follow the instructions as listed on the mysql webpage manual. Contact ECN to get it to be correctly located and not get wiped out every night. See "ASL Server" for information on giving Java programs access to the server.

The database was originally populated from flat files, processed by the java program 'sqlsetup.java'. This program does not need to be run, except to extend/modify the mysql database. This will only need to be done if the video files are changed.

Location data	
directory	/home/rv12/d/asldata/ASLdatabaseSearch/sqlsetup
source file	sqlsetup.java
schema file	tables.txt
data files	maintable.txt gloss.txt paragraph.txt subject.txt

The text file named `tables.txt` is a schema file and describes all of the tables that are used in the mysql database. The other text files are the tables.

To set up all of the tables, run the java file 'sqlsetup.class' to populate the mysql database.

```
cd ~/ASLdatabaseSearch/sqlsetup/  
java sqlsetup
```

Unless the database videos have changed, do not use this command.

3 The Client Side

Before beginning to test the client, it is good to start the ASL server. It is possible to tests some aspects of the client with just the ECN web server running, but to see query results the ASL server must also be running.

3.1 The Applet

The asldbsearch applet is a java program that runs on the client's machine. It provides a graphical user interface described in [1]. It communicates directly with the ASL Sever using the custom protocol described above. The asldbsearch applet runs within the client's browser; it redirects the browser to the results page once it is created.

Location data	
directory	/home/rvl2/d/asldata/public-web/
source file	asldbsearch.java
HTML file	asldbsearch.html

This is the entire contents of `asldbsearch.html`

```
<!DOCTYPE HTML PUBLIC>
<html>
<head>

<title>ASL DataBase Search</title>

<body>
<APPLET
CODE = "asldbsearch"
ARCHIVE = "asldbsearch.jar"
WIDTH = 1200
HEIGHT = 800 >
</APPLET>

</body>
</html>
```

3.1.1 Running the Applet

When a user accesses the database, the applet runs on their machine in their browser. During development, the application may additionally be tested as a stand-alone application or with the appletviewer program. These methods allow the program to be tested under circumstances where the first method is not operational. Most features of the applet, including connecting to the database and making queries, can be tested with all three methods. The applet must be run in a browser to test redirecting the browser to the final results.

To run the applet in the browser, guide the browser to `http://cobweb.ecn.purdue.edu/asldata/asldbsearch.html`

Type in the username and password. It is necessary to do this twice; once for the page and once for the applet.

To run the applet as a stand-alone application, use

```
cd ~/public-web
java asldbsearch
```

To run the applet with appletviewer, use

```
cd ~/public-web
appletviewer asldbsearch.html
```

Note again that neither the stand-alone application nor the appletviewer can redirect the browser, so neither of these methods can test viewing the results page.

3.1.2 Compiling the Applet

The applet must be compiled with Java 1.4, not Java 1.5. Many browsers do not support the more recent version.

The command for compiling the applet is

```
cd ~/public-web
javac -source 1.4 -target 1.4 asldbsearch.java
jar cvf asldbsearch.jar asldbsearch*.class
jarsigner -keystore rvlkeystore asldbsearch.jar rvlalias
password: rvl2asl
```

The last two of these commands pack the applet into a jar file and sign it. This allows us to open the socket connection. See the next section for more details.

3.1.3 Signing the Applet

To avoid socket connection security errors, we self-sign the applet. By default, java does not allow applets to make socket connections to outside hosts or ports. By signing the applet, we allow the user to grant the applet rights normally prohibited, including accessing the ASL server through a different port.

The following commands were used to create a self-signed key.

```
cd ~/ASLdatabaseSearch/server/
keytool -genkey -keystore rvlkeystore -alias rvlalias
(enter information requested such as name, address, etc.)
keytool -selfcert -keystore rvlkeystore -alias rvlalias
```

Once the key is created, it does not need to be re-created each time the applet is compiled. However, the applet does need to be placed into a jar file in order to be signed.

When using the signed version, the HTML code to points to the jar file instead of the class. The HTML code with the jar file is:

```
<APPLET
CODE = "asldbsearch"
ARCHIVE = "asldbsearch.jar"
WIDTH = 1200
HEIGHT = 800 >
</APPLET>
```

To not use the signed applet, change the code to refer directly to the class:

```
<APPLET
CODE = "asldbsearch.class"
WIDTH = 1200
HEIGHT = 800 >
</APPLET>
```

For general instructions on applet signing, visit these links

```
http://forum.java.sun.com/
thread.jspa?threadID=637377&messageID=3730635
http://java.sun.com/j2se/1.5.0/docs/guide/
plugin/developer_guide/rsa_signing.html
```

If the signed applet is not used, it is still possible for users to allow the connection. In this case, users must edit their java security settings in the `java.policy` file to include this line:

```
grant{
    permission java.net.SocketPermission
        "rvl2.ecn.purdue.edu:1024-",
        "accept, connect, listen, resolve";
};
```

It is necessary to re-start the browser for this change to take effect. This change is not required when the signed applet is used.

4 Automatic Restart

Occasionally the computer will be re-booted or one of the servers will disappear. We run a crontab script that checks the status of the programs periodically and re-starts them as necessary.

Location data	
maintainer	Robot Vision Lab
machine	rvl2.ecn.purdue.edu
mechanism	crontab
check script	/home/rvl2/d/asldata/bin/jsrvr.chk
ASL Server start script	/home/rvl2/d/asldata/bin/jsrvr.start
check interval	5 minutes
log files	
startup	/home/rvl2/d/asldata/JDBC/tmp/JDBCServer.log
ASL Server	/tmp/JDBCServer.out
MySQL Server	/tmp/mysqld.log

To edit the crontab file, use

```
bash # (Unless you use this by default)
export EDITOR=emacs
crontab -e
```

The crontab file contains a single line:

```
0 11 * * * * /home/rvl2/d/asldata/bin/jsrvr.chk
```

To forward any errors the crontab file makes to your personal web account, use

```
cd ~/
echo 'username@location.com' > .forward
```

All of this is documented in the `jsrvr.chk` file, as well:

```
/bin/jsrvr.chk
```

```
#!/bin/sh
#
# Shell Script for Starting the RVL-SLLL ASL Database Automatically
#
# Josiah Yoder
# This shell script is called from a script automatically, using crontab.
# To edit:
# export EDITOR=emacs
# crontab -e
# The current file contains exactly one line (and ends with a newline):
# 0 11 * * * /home/rvl2/d/asldata/bin/jsrvr.chk
#
# Things that did not seem to work:
# * Trying to detect using grep "java" | grep "asldata", because all processes use "asldata"
# * Trying to detect when the bash file is running using grep "jsrvr.start"
```

```

# (Even with the use | grep -v "grep")
#
#

# Check if server is alive, restart if necessary

PID='ps -Af | grep "java -classpath ./home/rv12/d/asldata/" | grep -v "grep"'
LOG=/home/rv12/d/asldata/JDBCServer.log

if [ ! -n "$PID" ]; then

## Pre-Apr 2008 version (replaced by the jsrvr.start call)
# java JDBCServer > /tmp/JDBCServer.out 2>&1 &

    /home/rv12/d/asldata/bin/jsrvr.start > /tmp/JDBCServer.out 2>&1 &
    date='/bin/date +%a %b %e %T'
    echo "$date: Server down. Restarted it." >> $LOG
else
    echo "$date: Server up." >> $LOG
fi

# Check if mysql is alive, restart if necessary

PID='pgrep "mysqld"'
LOG=/home/rv12/d/asldata/JDBCServer.log

if [ ! -n "$PID" ]; then
    /usr/local/libexec/mysqld --datadir=/home/rv12/a/mysql > /tmp/mysqld.log 2>&1 &
    date='/bin/date +%a %b %e %T'
    echo "$date: mysqld down. restarted it." >> $LOG
else
    echo "$date: mysqld up." >> $LOG
fi

```

The script for starting the ASL Server is

```
/bin/jsrvr.start
```

```

#!/bin/sh
cd /home/rv12/d/asldata/ASLdatabaseSearch/server;
java -classpath ./home/rv12/d/asldata/public-web/mysql-connector-java-3.1.10-bin.jar JDBCSE

```

5 Adding new users

We give each outside visitor a new username and password to the database once they sign the agreement available on the webpage. Ronnie creates a new username based on the last name of the registrant and a unique number for each user. The username is lastname_NNNN, where lastname is the family name and NNNN is the numeric ID of the user. The numeric IDs 1-9 are reserved for RVL-SLLL users.

For instance, if Joe Smith is user number 234, his ID will be smith_0234.

Each user is added manually with the following commands on asldata@rvl2.purdue.edu:

```
cd ~/passwd/  
htpasswd passwd.txt lastname\_NNNN
```

Users are asked to sign this agreement before receiving a username and password:

<http://cobweb.ecn.purdue.edu/RVL/Database/ASL/Agreement.pdf>

The usernames and encrypted passwords are stored in the file
~/passwd/passwd.txt

References

- [1] R. B. Wilbur and A. C. Kak, "Purdue RVL-SLLL American Sign Language Database," School of Electrical and Computer Engineering Technical Report, TR-06-12, 2006, Purdue University, W. Lafayette, IN 47906.
- [2] A. M. Martinez, R. B. Wilbur, R. Shay and A. C. Kak, "Purdue ASL Database for the Recognition of American Sign Language," Proceedings of IEEE International Conference on Multimodal Interfaces (ICMI), 2002.
- [3] Internal text file, asldata@rvl2.ecn.purdue.edu:README.txt