Remote Visualisation System

Installation Notes

Last modified: 14 March 2005

Introduction

This document contains instructions on building the RVS server and client components from source. There are some very specific prerequisites that need to be addressed before installing RVS so it is recommended that you follow the given instructions as closely as possible.

Supported Platforms

The following platforms have been succesfully tested using the intructions in this document:

- Debian Linux 2.4 (i386)

The RVS server comprises of multiple distributed components most of which are written in Java, making them platform independent. Although the server has been designed with Linux in mind, the Java components should compile and run on other Java supported platforms.

The Session Invocation Service (SIS) component has been developed in C++ and tested on the platforms listed above.

Software Requirements

The following software is required by RVS during the build process as well as at runtime. They are not distributed with RVS.

NOTE: The software versions shown below are the ones that the RVS has been tested and known to work with.

Java

- Java v1.4.* (http://java.sun.com/j2se)

  Debian package: Yes

  Redhat package: Yes

- Apache Tomcat v4.* or higher (http://jakarta.apache.org/tomcat)

  Debian package: Yes

  Redhat package: Yes

- Doxygen v1.3.6 or higher (www.doxygen.org)

  Debian package: Yes

  Redhat package: Yes
- JacORB v2.0 (http://www.jacorb.org)

  Debian package: No

  Redhat package: Yes (http://www.jpackage.org)
- Xerces Java v2.6.2 (http://xml.apache.org/xerces2-j/index.html)

  Debian package: Yes

  Redhat package: Yes (http://www.jpackage.org)
- Proguard v2.1 (http://proguard.sourceforge.net)

  Debian package: No

  Redhat package: No
- Apache Ant v1.6.1 or higher (http://ant.apache.org)

  Debian package: Yes

  Redhat package: No
- KeyPoint PNG Encoder v1.5 (http://catcode.com/pngencoder)

  Debian package: No

  Redhat package: No

Other Software

- AIPS++ v19.8* or higher (http://www.atnf.csiro.au/computing/software/aips++/release/docs/aips++.html)

  Debian package: No

  Redhat package: No
- ACE v5.3.5 and TAO v1.3.5 – bundled (http://www.cs.wustl.edu/~schmidt/TAO.html)

  Debian package: No

  Redhat package: No
- Xerces C++ v2.4.0 (http://xml.apache.org/xerces-c)

  Debian package: Yes

  Redhat package: Yes
- Gcc 3.2.*. This is the only version that has been tested. It is very likely that newer versions will also work. NOTE: All C/C++ source must be built using the same version of gcc.

  Debian package: Yes

  Redhat package: Yes

Obtaining RVS

You can download RVS binary and source distributions as gzipped tarballs or anonymous CVS from http://www.atnf.csiro.au/vo/rvs.

Environment Setup

The RVS build process requires you to set up some environment variables and edit configuration files that will be used during the process. These are listed below:

Set RVS_HOME

Set your RVS_HOME environment variable to the root of your RVS tree. e.g, RVS_HOME=/usr/local/software/rvs.

Set CATALINA_HOME

Set your CATALINA_HOME environment variable to the root of your Tomcat web server installation. e.g, CATALINA_HOME=/usr/local/software/jakarta-tomcat-5.0.28.

Rename config files

If you are using a CVS distribution, you will need to remove the '.template' suffix from all config files in $RVS_HOME/config. You can do this using the 'mv' command on all files. For example,

```
shell% cd $RVS_HOME/config && mv rvsconfig.xml.properties rvsconfig.xml
```

Edit build.properties

The $RVS_HOME/config/build.properties file contains configuration information used by Ant when building Java components. Edit the properties to reflect your environment.

Edit rvsenv.incl

The $RVS_HOME/config/rvsenv.incl file contains configuration information used by RVS at build time and also at runtime. You will notice that these Edit the properties to reflect your environment.

Edit client.conf

The $RVS_HOME/config/client.conf file contains configuration information used by RVS clients. It contains paths and URLs to the web services that the client communicates with.

Edit rvsconfig.xml

The $RVS_HOME/config/rvsconfig.xml file contains runtime configuration parameters about the RVS Server.

Edit jacorb.properties

The $RVS_HOME/config/jacorb.properties filecontains runtime configuration information. You only need to modify the “ORBInitRef.NameService” property at this stage. Change the 'domain' part of the url so it points to the server where you're hosting the RVS web service.

Copy jacorb.properties to $HOME

The jacorb.properties file mentioned above needs to be accessed from the **$HOME directory of the user that starts Tomcat**. It is advisable that you create a symbolic link to the file in the $HOME directory. For example:

```
shell% ln -s $RVS_HOME/config/jacorb.properties $HOME
```

Deploy Apache Axis

The axis web application is need by the RVS web service. To deploy the webapp, simply copy $AXIS_HOME/webapps/axis (where AXIS_HOME is the axis installation directory) to $CATALINA_HOME/webapps.

Building RVS Server Components

There are two different build environments for the RVS server; one for all java components and another for the C++ component.

Generate CORBA Stubs & Skeletons

Before you compile and RVS source you need to generate the CORBA stubs and skeletons that are defined by idl interfaces. You can do this by runing the 'buildidl.sh' script:

```
shell% $RVS_HOME/bin/buildidl.sh
```

If you get errors, ensure that the paths in $RVS_HOME/config/rvsenv.incl are set correctly.

Java Components

The Java components consists of several back-end executable components and some web applications that can be deployed to your Tomcat web server. To compile the RVS server java components and deploy the web applications, follow the instructions below.

Confirm build properties

If you haven't done so already, you need to edit the $RVS_HOME/config/build.properties as mentioned above, in 'Environment Setup'. You are likely to get errors if this is not done correctly.

Compile all components

Go to $RVS_HOME/src/java and run ant. e.g:

```
shell% cd $RVS_HOME/src/java

shell% ant
```

If you get errors, you will need to use the error messages to deduce the cause. In most cases, errors will be related to libraries that ant cannot find.

Ant should have generated the following files/directories:

- $RVS_HOME/lib/aips.jar – classes needed for interfacing with AIPS++ via CORBA
- $RVS_HOME/lib/rvs.jar – all other implemented classes in RVS.
- $RVS_HOME/lib/rvsservice.war – the web application archive for the RVS web service.
- $RVS_HOME/webapps – contains the rvsservice web application.

Deploying the RVS Web Service

The RVS Web Service is contained in the rvsservice web application. There are a couple of ways of deploying the service. The recommended method is to create a symbolic link in $CATALINA_HOME/webapps/ pointing to $RVS_HOME/webapps/rvsservice/. e.g:

```
shell% cd $CATALINA_HOME/webapps

shell% ln -s $RVS_HOME/webapps/rvsservice
```

The other method is to deploy a copy of the webapp (not linked to $RVS_HOME/webapps/rvsservice). e.g:

```
shell% cp -rf $RVS_HOME/webapp/rvsservice $CATALINA_HOME/webapps
```

or

```
shell% cp RVS_HOME/lib/rvsservice.war $CATALINA_HOME/webapps
```

NOTE: If you use the copy method then you need to remember to deploy the webapp after each time you compile the server.

C++ Components

In some cases the TAO libraries (*.so files) may not be placed in a linker-accessible path, your $LD_LIBRARY_PATH. They exist in the source directory in which they were built. It is required that links to these libraries are created inside one of the shared directories in the $LD_LIBRARY or inside ($RVS_HOME/lib/linux). The latter can be done as follows.

```
shell% ln -s `find $TAO_ROOT -name \*.so` $RVS_HOME/lib/linux
```

where TAO_ROOT is the root directory of your TAO installation. You also need to have your AIPS++ environment set. This is typically done by sourcing the aipsinit file in your aips++ home directory. e.g:

```
shell% source /myapps/aips++/aipsinit.csh
```

The source can then be built as follows.

```
shell% cd $RVS_HOME/src/cpp

shell% ./makecorba.sh

shell% make

shell% make SISServer -C sessionmgr
```

The above process should have created the executable $RVS_HOME/bin/linux/SISServer. If this file was not created, there must have been an error in the build process. Try to deduce the cause via any error messages.

Building Client Components

You may build the client components as follows:

```
shell% cd $RVS_HOME/src/java/rvs/client

shell% ant rvsviewer.package

shell% ant clientutils.package
```

The two 'ant' commands above will create two 'war' files inside $RVS_HOME/lib; rvsviewer.war (NOT rvsservice.war, which is the server application) and clientutils.war. Both these web applications need to be deployed to your web server. There are two ways of doing this in Tomcat (version 4.0 or higher):

1. Copy the two files to $CATALINA_HOME/webapps and (re)start the server.
2. Use the Tomcat Manager via the web browser. For help on using the manager, go to http://jakarta.apache.org/tomcat/tomcat-4.0-doc/manager-howto.html.

Test it!

That's it!! Use the RVS Server User Guide for instructions on starting/stopping the server. You can then run the RVS Viewer client to test it out.

```
shell% java -jar $RVS_HOME/lib/rvsviewer.jar
```