# AgentTeamwork: Enhancing the Resource Agent's Features

## An Interim Report

**Undergraduate Research Project (Summer 2004)**
**Student: Shane Rai**
**Advisor: Prof M Fukuda**

## Issues Running Xindice Database Server

From the very beginning of installing Xindice on Medusa, I came across a host of problems in getting this database server to run. The following is a chronological series of issues that I faced while trying to get the server up and running.

→Xindice uses port 4080
Upon running Xindice for the first time, the following run time errors were generated:

ERROR: Could not start service 'HTTPServer'
Service: 'APIService' started
FATAL ERROR: Service manager could not be started

At first thought, my impression was perhaps port 4080 was not open on Medusa. The CSS Systems Admin (Doug McLean) was requested to open this port. Turns out that port 4080 was not open on Medusa and the firewall was accordingly adjusted. However, even after that, this same error persisted leading me to the conclusion that perhaps this was not a port related issue.

→No web server installed on Medusa
After consulting with Doug, we came to the conclusion that the absence of a web server on Medusa could be responsible for this exception. This conclusion, of course, was made because of the nature of the exception that was being thrown viz. HTTPServer service failing to start. In any case, we ended up installing Tomcat on Medusa as the web server following Prof Fukuda's suggestion.

→Issues in restarting Xindice
After Tomcat was installed, I was able to successfully start Xindice. However, I could not get the server to shut down the way it was documented to be. When a subsequent restart was attempted on the server, the same errors as above resurfaced. It turned out that since Tomcat was running as the web server and Xindice was not shut down properly, the port that Xindice was using (when it started) was still in use and hence restarting the database on the same port was resulting in that exception.

→No root access to Medusa
It seems that Doug, having root access to Medusa, successfully managed to get Xindice up and running when he installed the same on Medusa's root folder (under /usr/local/xindice). However, since root privileges cannot be given to any of the student project members, it seemed that there was no way to run Xindice without such access rights (given these problems). The end result of all this was to consider another native XML database that would support the requirements of the project.

## *eXist as the new XML database sever*

Eventually, a similar open source native XML database was found that so far has not resulted in any major exceptions that have proved to be hard to debug. This database is called eXist. For more information, visit the site: http://exist.sourceforge.net/.

### Running the database

eXist offers three ways in which the server can be started. It could be run either as a standalone server process, embedded into an application (such as Java) or used in connection with a servlet engine. For the purpose of this project, the last option is not needed.

### Running the server as a standalone process

In this case, the server runs inside its own Java Virtual Machine. The HTTP listener runs on port 8088 and XMLRPC on port 8081. Clients have access to the server using either of these protocols.

*Starting the server*
Follow the steps to start the server as a stand alone process. I assume that the root folder for the installation is named eXist. Open a terminal window and enter the following commands.

→cd eXist/bin
→./server.sh

The server should start and the following lines should be displayed at the end of the dump:
……..
……..
starting HTTP listener at port 8088
starting XMLRPC listener at port 8081

*To shutdown the server*
Open another terminal window. Enter the following commands:

→cd eXist
→java -jar start.jar shutdown --uri=xmldb:exist://localhost:8081

The server should shut down. Also look at the other terminal window that was used to start the server. That window should also print some messages about shutting down and killing all client threads.

*Using the client tool*
eXist has a neat little Java client interface to the database. To start this client, open a terminal window and enter the following commands:

→cd eXist/bin

→./client.sh

A log in dialog box should appear; leave the password box blank; press OK. The eXist Admin Client window should open; in the upper pane, you can view the existing collections in this database; the bottom pane acts like a command line interface allowing the user to enter some specific commands (check out the online doc for this syntax).

To exit from this client, type "quit" at this command line to close the client.

NOTE: I have had issues trying to run this client remotely via SSH. I have not been to conclude the reasons for this.

## Running the server as an embedded instance

For this project, the database will be used as an embedded instance. In this case, the database is started in the same JVM as that of the client application. The database will not be accessible from outside this client app. This would be the preferred way to running the server for this project since it does not require any user intervention to start and shutdown the server (although this process could be handled inside the client app). In any case, the embedded instance can make use of the XML:DB API to programmatically start and stop the server.

*The XML:DB API*
According to the online documentation, it is suggested that using XML:DB API is the preferred way to work with the database from client apps. These applications are to be written in Java.

Quote: “The XML:DB API provides a common interface to native or XML-enabled databases and supports the development of portable, reusable applications.” For a quick tour on how to use this API for common database tasks, visit http://exist.sourceforge.net/devguide.html#N102EE

*A Java Client app to access the Database*
The following is the complete source of a simple Java client that is used to start an embedded instance of the database. A test collection was added prior to the database and is called “Test”. An XML document was also added to this collection prior. In this example, this same XML document is being retrieved using its resource ID which was generated when the document was added to the

database. The contents of this XML document are then printed to the standard output.

This example has been modified from an example that is available in the installation samples.

```
/* A demo Java client app instantiating an embedded instance of eXist */

import org.exist.xmldb.DatabaseInstanceManager;
import org.xmldb.api.DatabaseManager;
import org.xmldb.api.base.*;
import org.xmldb.api.modules.*;
import org.xmldb.api.*;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.OutputKeys;
import org.exist.storage.serializers.EXistOutputKeys;

public class Retrieve {

        protected static String driver = "org.exist.xmldb.DatabaseImpl";
        protected static String URI = "xmldb:exist:///db/test";


        public static void main(String args[]) throws Exception {

                // initialize database drivers
                Class cl = Class.forName(driver);
                Database database = (Database) cl.newInstance();
                database.setProperty("create-database", "true");
                DatabaseManager.registerDatabase(database);

                // get the collection
                try {
                        Collection col = DatabaseManager.getCollection(URI,
                        "admin", "");
                        if (col == null)
                                System.out.println("col is null\n");

                        col.setProperty(OutputKeys.INDENT, "yes");

                        col.setProperty(EXistOutputKeys.EXPAND_XINCLUDES
                        , "no");

                        col.setProperty(EXistOutputKeys.PROCESS_XSL_PI,
                        "yes");
```

```
                XMLResource res =
                        (XMLResource)col.getResource("7b44705f.xml");

                if(res == null)
                        System.out.println("document not found!");
                else
                        System.out.println(res.getContent());

        DatabaseInstanceManager manager = (DatabaseInstanceManager)
                col.getService("DatabaseInstanceManager", "1.0");
                manager.shutdown();
        }
        catch (XMLDBException e) {
                        System.err.println("XML:DB Exception occured "
                        + e.errorCode);
        }
    }
}
```