# Resource Agent & Commander Agent Reference Doc

## Project AgentTeamwork

**Author:** Shane Rai

**Advisor:** Prof M Fukuda

**Class:** CSS 499 (Undergrad Research Project)

## Table of Contents

## XML Database (eXist)

This section briefly describes how this XML database is used in the context of AgentTeamwork. For more detailed information on eXist, please refer to the Resources section at the end of this document.

### a. XML Database Concepts

When writing Java apps that accesses eXist, it is important to get an understanding on how XML databases are organized. The concept of drivers, collections, resources and services are important. Refer to the Resources section of this document under "About Writing Java Apps with XML DB API". You will find some very useful boiler plate code that is illustrated there. Such code also is available in \samples\org\exist\examples\xmldb under the root folder where you install eXist. Many other sample codes are also present along this path. However, in our case we will be mostly concerned with XML DB sample code.

### b. Server Deployment

One of the interesting features of eXist is the possibility of running the database in three alternatives. It may run either as a standalone process whereby it runs in its own Java Virtual Machine. It may also be embedded into an application whereby it is controlled by this app. It basically runs in the same JVM as the app itself. The client app will have full access to the database. And lastly, it may run in a servlet context.
For this project, it was decided that eXist will be used in the embedded instance. The reason is that the database need not necessarily be accessed through the network but instead will reside on the same machine as the community user. Hence, the AgentTeamwork app itself will access the database whenever it is needed.

### c. The XML DB API

According to the documentation of the server, it has been suggested that the preferred way to access the database from Java apps is via the XML DP API. This API provides a common interface to native XML databases such as eXist. This XML database has its own implementation of this interface. This is what has been used in Resource Agent.

# Resource Agent API

The following section describes the Application Programming Interface of the ResourceAgent class.

**<u>Data Members</u>**

*<u>Note</u>: Only the "major" data members of this class have been enumerated here. Some of the others are common across all other Agent class definitions.*

| | |
|---|---|
| Member: | private Hashtable rscArgsTable |
| Description: | hash table that holds all the resource requirements of this task |
| | |
| Member: | private int numRqdArgs = 5 |
| Description: | number of required arguments to construct a ResourceAgent object (set to 5) |
| | |
| Member: | private transient Collection rsc_Col = null |
| Description: | represents an instance of the resource collection that is stored in the local XML database server |
| | |
| Member: | private FTPClient ftp |
| Description: | represents an instance of the FTP server where the XML resources are stored |
| | |
| Member: | private boolean isConnectedToFTPServer = false |
| Description: | true if this ResourceAgent is connected to the FTP server, false otherwise |
| | |
| Member: | private String rscXMLDir = "ResourceXML" |
| Description: | name of the folder on the FTP site where the XML resource files reside |
| | |
| Member: | private String osName |
| Description: | operating system where this ResourceAgent is executing (see Design Suggestions section in this doc for how this info can be utilized) |
| | |
| Member: | private Timer probeTaskTimer |
| Description: | a timer class to allow for periodic execution of ResourceAgent's tasks (see Design Suggestions for more info) |
| | |
| Member: | private class remoteRscProbeTask |
| Extends: | class TimerTask |
| Description: | class whose run( ) method that will be responsible performing all the periodic tasks required of the ResourceAgent (see Design Suggestions and Unresolved Issues for more info) |

**Methods**

*Note: Only those methods that are specific to this agent have been enumerated here. The other methods are common across all other Agent class definitions. However, some of those common methods have also been listed here.*

| | |
|---|---|
| Function: | ResourceAgent (String[] args) |
| Description: | Constructor |
| Arguments: | args : array containing all the resource requirements for this task |
| Calls: | getAllConstrArgs |
| Return: | void |

| | |
|---|---|
| Function: | getAllConstrArgs (String[] args) |
| Description: | adds all the required constructor requirements to rscArgsTable from args |
| Arguments: | args : array containing all the resource requirements for this task |
| Calls: | N/A |
| Return: | void |

| | |
|---|---|
| Function: | init ( ) |
| Description: | initializes this ResourceAgent and also starts a Thread to receive all incoming messages from other agents |
| Arguments: | none |
| Calls: | initDB( ) and mainMethod( ) |
| Return: | void |

| | |
|---|---|
| Function: | initDB ( ) |
| Description: | starts an embedded instance of the XML database (eXist) on the local machine. Checks to see if a Collection called "resources" exists in t he database. If not, creates one. |
| Arguments: | none |
| Calls: | the XMLDB API of eXist |
| Return: | void |
| Note: | this is mostly boiler plate code and has been copied from the sample code available with the database download |

| | |
|---|---|
| Function: | mainMethod ( ) |
| Description: | performs operations such as updating the XML database with the latest resource XML files and starting the timer for executing the periodic tasks. The main thread is put to sleep while the main thread's data member (main_thread_cont) is true. Also responsible for calling the function to stop the message thread when this data member is set to false i.e. the main thread needs to exit. |

| | |
|---|---|
| Arguments: | none |
| Calls: | updateDB( ), scheduleProbingTasks( ) and stopSubThreads( ) |
| Return: | void |

| | |
|---|---|
| Function: | updateDB ( ) |
| Description: | responsible for calling functions that connect to the remote FTP server, compares the resource XML files on the server with the local database and downloads those resources that are either have a newer copy on the FTP server or not present on the local database. |
| Arguments: | none |
| Calls: | connectToFTPServer ( ), getResourcesListFromFTPServer ( ), downLoadXMLResourcesFromFTPServer ( ) and getResourcesListFromDatabase( ). |
| Return: | true if all operations are executed without any exceptions, false if any operation fails. |

| | |
|---|---|
| Function: | connectToFTPServer ( ) |
| Description: | responsible for connecting to the remote FTP server where the resource XML files are uploaded by the community members. |
| Arguments: | none |
| Calls: | Commons FTP API |
| Return: | isConnectedToFTPServer i.e. true if the Agent got connected to the FTP server (or was connected in the first place) and false if failed to connect. |

| | |
|---|---|
| Function: | getResourcesListFromFTPServer ( ) |
| Description: | responsible for obtaining a list of XML resource files that are stored on the FTP server |
| Arguments: | none |
| Calls: | Commons FTP API for getting a list of the files needed |
| Return: | array of FTPFile where each element is the name of the resource file on the FTP server |

| | |
|---|---|
| Function: | getResourcesListFromDatabase ( ) |
| Description: | responsible for obtaining a list of XML resource files that are stored on the local XML database's resources Collection |
| Arguments: | none |
| Calls: | XML DB API for getting a list of the files stored |
| Return: | a vector of XML resource names that reside in the local database's resources Collection |

| | |
|---|---|
| Function: | downLoadXMLResourcesFromFTPServer ( ) |
| Description: | responsible for comparing the last modified date of the XML resource files on the FTP server and local database and downloading those resources that are either newer on FTP site or not available on the database |
| Arguments: | FTPFile[] ftpRscXMLFilesList: array of XML resources on the FTP server<br>Vector dbRscXMLFilesList: array of XML resources on the local database |
| Calls: | addResourceFromFtpServerToDB( ) to download XML resource from FTp server to local database |
| Return: | true if all operations executed successfully else false if any exception occurred |

| | |
|---|---|
| Function: | addResourceFromFtpServerToDB ( ) |
| Description: | responsible for performing the download of ftpResourceFile to the local machine and adding it to the local XML database |
| Arguments: | String ftpResourceFile: the XML resource file to download from the FTP server |
| Calls: | XML DB API and Commons FTP API |
| Return: | void |

| | |
|---|---|
| Function: | scheduleProbingTasks ( ) |
| Description: | responsible for creating a Timer class instance and scheduling it with an instance of the remoteRscProbeTask class. The latter class' run( ) method will perform the needed periodic operations that this agent needs to execute. |
| Arguments: | none |
| Calls: | schedule method of the Timer class instance |
| Return: | void |
| **Note:** | Currently, the following run time error occurs when the schedule method is called of the Timer class' instance:<br>`UWPlace#run : java.lang.reflect.InvocationTargetException agent name: ResourceAgent Cause : java.lang.IllegalAccessError: tried to access class ResourceAgent$remoteRscProbeTask from class ResourceAgent`<br>→Please refer to the Unresolved Issues section of this document for some possible workarounds. |

## Commander Agent API

Please note that only a part of this agent's source was changed so as to implement the features of the Resource Agent. Only those methods and data members that were added for that purpose have been documented in this section.

### <u>Data Members</u>

| | |
|---|---|
| Member: | private int numRqdArgs = 11 |
| Description: | the number of required arguments to construct this agent |

| | |
|---|---|
| Member: | private Hashtable rscHashTable |
| Description: | Hashtable that contains all the resource requirements for the current task |
| Note: | THIS IS NOT NEEDED ANYMORE; PLEASE SEE NEXT DATA MEMBER |

| | |
|---|---|
| Member: | private String[] rscAgentArgs |
| Description: | string array containing all the resource requirements for the current task |

| | |
|---|---|
| Member: | private List rscAgentIds = new ArrayList() |
| Description: | a list of id's of all resource agents spawned by this Commander Agent |

### <u>Methods</u>

| | |
|---|---|
| Function: | CommanderAgent (String[] args) |
| Description: | this agent's constructor where the resource agent's arguments are parsed, the user program is extracted along with its arguments as well as required classes |
| Arguments: | String [] args: arguments needed to construct this agent |
| Calls: | getAllRscTypes( ) (this function is no longer needed), usage( ) |
| Return: | void |

| | |
|---|---|
| Function: | getAllRscTypes ( ) |
| Description: | THIS METHOD IS NOT NEEDED ANY MORE |
| Arguments: | N/A |
| Calls: | N/A |
| Return: | N/A |

| | |
|---|---|
| Function: | init ( ) |
| Description: | this agent's init method where (similar to other agent's init method) initialization occurs. *Code that was added was the call to spawnInitAgents( ).* |
| Arguments: | none |

| | |
|---|---|
| Calls: | spawnInitAgents ( ) in addition to the previously existing calls to other methods |
| Return: | void |

| | |
|---|---|
| Function: | spawnInitAgents () |
| Description: | responsible for spawning the agents that are needed during initialization of this agent such as spawning the Resource Agent |
| Arguments: | none |
| Calls: | spawnChildAtPlace (String destHost, String type, String[] args) |
| Return: | void |

| | |
|---|---|
| Function: | spawnChildAtPlace (String destHost, String type, String[] args) |
| Description: | responsible for spawning the agent specified by type with its arguments given by args at destHost. *This method was edited to include the switch for spawning the Resource Agent.* |
| Arguments: | String destHost: location to spawn the agent |
| | String type: type of agent to spawn |
| | String[] args:   arguments of the spawned agent |
| Calls: | UWAgent.spawnChild (String type, String[] args, String destHost) and other methods |
| Return: | void |

| | |
|---|---|
| Function: | respondToMessage(UWMessage message) |
| Description: | responsible for deciding what action is to be taken based on the UWMessage message recieved by this agent. *Please note that messages from Resource Agent will have to be added here. Only one message ("rsc_itenarary_for_new_task") has been added here but it does not have any corresponding response action.* |
| Arguments: | UWMessage message: the message received by this agent |
| Calls: | functions depending on message |
| Return: | void |

## Starting AgentTeamwork (A Demo Setup)

This section briefly describes, via an example, the procedures needed to be followed to start the program.

### a. Injecting Commander Agent

The following command line input is needed:

```
java UWInject localhost CommanderAgent -u "C:\Documents and Settings\Shane R\My Documents\CSSClasses\CSS499(UnderGradResearch)\Summer2004\agents" -p UWPlace -n 35353 -c myClient -m 100 ip 123 cpu_speed 456 cpu_count 5 memory 45 os win disk 45 total 5 time 0 cpu_arch win rscAgent ftp.tripod.com$agentTeamWork$test$0.125$C:\eXist prog shane$arg1$arg2$arg3 r class1$class2$class3
```

Please refer to Prof Fukuda's documentation on the "original" arguments for this agent. Arguments that were added as a result of the Resource Agent's resource requirements are as follows:

```
rscAgent ftp.tripod.com$agentTeamWork$test$0.125$C:\eXist
```

where:
→ftp.tripod.com is the FTP server being used to store the community users XML resource files
→ agentTeamWork is the user name to log into this FTP server
→test is the password for the above user name
→0.125 is the probe frequency in minutes
→C:\eXist is the location where the XML database eXist was installed

### b. Starting RMI Server

The RMI server needs to be started on another command window at the port 35353 since the Commander Agent was injected at the same port (look at above inject command's –n switch). Type out the following command:

```
rmiregistry 35353
```

### c. Starting UWPlace

A UWPlace also needs to be started on another command window. Simply type in the command:

```
java UWPlace
```

## d. Setting CLASSPATH

This Java environment variable need to be set correctly otherwise numerous run-time or compile time errors might be generated. Below is a snapshot of what CLASSPATH might look like. Please note that this example is valid for a particular machine. Use it as a guideline as to which JAR files are needed to setup the infrastructure.

```
.C:\Program
Files\Java\j2re1.4.2_05\bin;C:\eXist\exist.jar;C:\eXist\lib\core\
xmldb.jar;C:\eXist\lib\core\resolver-
20030708.jar;C:\eXist\lib\core\jakarta-oro-
2.0.6.jar;C:\eXist\lib\core\antlr.jar;C:\eXist\lib\core\xmlrpc-
1.2.jar;C:\eXist\lib\core\commons-pool-
1.1.jar;C:\eXist\lib\endorsed\xerces-
2.6.1.jar;C:\eXist\lib\endorsed\xalan-
2.5.2.jar;C:\eXist\lib\endorsed\xml-
apis.jar;C:\eXist\lib\core\log4j.jar;C:\eXist\commons-httpclient-
2.0.1\commons-httpclient-2.0.1.jar;c:\eXist\lib\core\commons-net-
1.2.2.jar;C:\Documents   and   Settings\Shane   R\My   Documents\CSS
Classes\CSS499(UnderGradResearch)\Summer2004\UWAgent\UWAgent.jar;
C:\Documents     and     Settings\Shane     R\My     Documents\CSS
Classes\CSS499(UnderGradResearch)\Summer2004\GridTcp\GridTcp.jar;
C:\Documents     and     Settings\Shane     R\My     Documents\CSS
Classes\CSS499(UnderGradResearch)\Summer2004\agents;C:\Documents
and          Settings\Shane          R\My          Documents\CSS
Classes\CSS499(UnderGradResearch)\Summer2004\UWAgent;c:\eXist
```

*Please note that some JAR files might not be needed but were added for possible future need. Also note that the JAR file names might also be different for a newer distribution.*

## e. JAR Files

The JAR files that are needed are those relating to the XML database (eXist), FTP client and of course, the AgentTeamwork JAR files as well.

The JAR files of the database are mostly found in lib\core directory off the root folder of the database installation. Some are also found in lib\endorsed.

The FTP client related JAR file (commons-net-1.2.2.jar) is also found in lib\core. If it is not, it can be found on: http://jakarta.apache.org/site/binindex.cgi
Scroll down till you find Commons Net.

The JAR files for AgentTeamwork can be downloaded off Medusa.

## Design Suggestions & Tips

This section briefly discusses some suggestions that can be utilized to implement those requirements of the Resource Agent which remain to be implemented.

### a. Probing Remote Computers

For the Resource Agent to add a remote node's actual resource availability (such as available disk space, available memory, etc) to its copy on the local database, this agent will need to get this information from potentially different operating systems of the community members. One solution would be to spawn a child of the parent Resource Agent for each remote computer. Each of these child agents, based on the OS of the remote machine (osName data member of ResourceAgent), will execute OS specific commands to get the resource information. E.g. to get a memory snapshot on a Windows machine, you can use the "mem" DOS command. Of course, the output will have to be parsed to extract the relevant resource information.

Hence, a parser class might have to be written that considers the output of each resource information that is needed for each platform.

For some information on relevant Linux commands, check: http://www.sibbald.com/linux/admin01.html

Furthermore, to check the network availability of a remote node, the ping utility can be used for this purpose. However, once again a parser needs to be written for each OS to determine this availability. For example,

*On a UNIX machine:*

```
ping -c 5 -q -W 10 192.168.0.100
```

where:
-c is the number of packets to send
-q is quiet mode
-W is timeout in seconds

The subset of the output that is of importance is:

```
--- 192.168.0.100 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 3999ms
```

*On a Windows machine:*

```
ping -n 5 127.0.0.1
```

where:
-n is number of packers to send

A demo program called **Ping1.java** has been written that briefly demonstrates this parsing. It is located on /home/uwagent/shane/Proof_Of_Concept on Medusa.

## b. The Timer and TimerTask Classes

These classes can be used for executing the periodic tasks of probing the remote computers and updating the availability of those nodes that needs to be performed by the Resource Agent.

In a nutshell, you need to extend the TimerTask class and this new subclass' run() method will be responsible for executing the periodic tasks. Essentially, a separate thread is spawn for this task.

For more information on how to use these classes, check out sample code at:
http://java.sun.com/docs/books/tutorial/essential/threads/timer.html

*Please note that a run time error currently takes place:*

```
UWPlace#run : java.lang.reflect.InvocationTargetException
agent name: ResourceAgent
Cause  :  java.lang.IllegalAccessError:  tried  to  access  class
ResourceAgent$remoteRscProbeTask from class ResourceAgent
```

The root cause seems to be *class remoteRscProbeTask*. Perhaps this class needs to be moved elsewhere and not be kept nested. This needs further investigation.

## c. Using eXist Community Forum

The XML database (eXist) has a somewhat resourceful forum to get your questions answered. The author of this program, Wolfgang Meier, is quite prompt in getting back to one's queries. Use this resource freely since the very fact that this program is open source, bugs or limited documentation might be common.

Queries can be emailed to: exist-open@lists.sourceforge.net

## d. FTP Server Directory as System Property

Instead of specifying the Resource XML directory on the FTP server as a data member of the ResourceAgent class (private String rscXMLDir = "ResourceXML"), perhaps this could be set as a System property of the application itself. From a architectural point of view, this is more elegant and feasible if in the future some System properties need to be read in from a file (for example).
This can be achieved using the API System.setProperty(…).

## e. Handling Exceptions

An exception handling framework along with rules needs to be defined as to how exceptions need to be handled by the various agents and other utility classes. For example, how will exceptions be handled higher up in the code and under what circumstances will the application need to bail from the current Java Virtual Machine. Such a framework will bring about a unified, consistent and predictable approach to exception handling.

## f. Super Class for All Agents

Currently, the class UWAgent serves as a super class for all other agents derived from it. However, there are many methods whose code is duplicated across many agents. For example, the method init( ) of most agents typically starts a message retrieval thread. Such methods can be easily derived as well from UWAgent. This thinking also extends to the common data members of the derived as well as UWAgent.

## g. Abstracting Central Repository

Currently, an FTP server (ftp.tripod.com) is used to store all the XML resource documents of all the community members. However, other repositories types that could be used by other community groups include an internet group, HTTP server, etc. Hence, it might be efficient if an abstract class can be written that will abstract all the logic for accessing such repositories for downloading all the XML resources.

## Resources

The following is a list of web resources that might be useful for further enhancing the agent's requirements.

### a. eXist Home Page

The XML database's home page can be located at: http://exist.sourceforge.net/

### b. JavaDocs for eXist (XML DB API)

The API documentation for using the XML DB interface of the database can be found at: http://exist.sourceforge.net/api/index.html

### c. About Writing Java Apps with XML DB API

To get an understanding on how to write Java apps that query and access the XML database, refer to:
http://exist.sourceforge.net/devguide.html#N10315

### d. eXist Server Deployment

This XML database can be deployed under three different scenarios. For information on these, refer to:
http://exist.sourceforge.net/deployment.html

Since we are concerned with the embedded instance, refer to in particular:
http://exist.sourceforge.net/deployment.html#N101DA

### e. eXist Forum

A forum exists which at times is useful to search for solutions to any issues. This is located at: http://sourceforge.net/mailarchive/forum.php?forum_id=3154

### f. Commons Net FTP JavaDocs

The API for this package is located at:
http://jakarta.apache.org/commons/net/apidocs/org/apache/commons/net/ftp/FTPClient.html#listFiles(java.lang.String)
This is needed to for using the FTPClient class.