

Cancer Trials Support Unit

CTSU – A Service of the National Cancer Institute

RandoNode Version 3.0 Readme Document

Revision 5

30 July 2014

Table of Contents

1	INTRODUCTION	4
1.1	OVERVIEW	4
1.2	AUDIENCE	4
2	GENERIC CONFIGURATION SET UP FOR RANDONODE.....	4
2.1	CONFIGURE \$CTSU_HOME VARIABLE	4
2.2	HOW TO SETUP DEMOGRAPHIC PUBLIC ID MAPPING FILE.....	4
2.3	HOW TO INSTALL META DATA FILES	5
3	PACKAGES / NAMESPACES CONTAINED WITHIN THE RANDONODE STARTER KIT.....	6
4	STEP BY STEP PROCESS FOR JAVA RANDONODE	6
4.1	HOW TO SETUP RANDONODE CONFIGURATION FILE	6
4.2	HOW TO INSTALL RANDONODE.JAR FILE OR RANDONODE STARTER KIT CODE	7
4.2.1	<i>Details of RandoNode.jar</i>	<i>7</i>
4.2.2	<i>Installation/Upgrade Steps</i>	<i>7</i>
4.2.2.1	Integrate RandoNode.jar file	8
4.2.2.2	Integrate RandoNode starter kit source code.....	8
4.3	HOW TO BUILD THE PROJECT AND GENERATE RANDONODE.WAR FILE	8
4.4	HOW TO TEST THE LATEST RANDONODE INSTALLED.....	9
5	STEP BY STEP PROCESS FOR .NET RANDONODE.....	12
5.1	HOW TO UPGRADE EXISTING RANDONODE	13
5.2	HOW TO TEST THE LATEST RANDONODE	13

1 Introduction

1.1 Overview

This document provides instruction to developers on how to install RandoNode Starter Kit version 3.0 or upgrade existing RandoNode to RandoNode Starter Kit version 3.0 and integrate their code with the Starter Kit.

This document covers installation/upgrade steps for both the Java and .NET RandoNode starter kits. Section 3 contains the details of Java RandoNode, and section 4 contains the details of .NET RandoNode.

1.2 Audience

The audience of this document is the IT resources at LPOs and CTSU.

2 Generic Configuration Set Up for RandoNode

This section describes the configuration needed to run the RandoNode. These are one time configurations needed during RandoNode installation, except configuring metadata file. No configuration change is needed for upgrade. Metadata file configuration is at the protocol level. LPO needs to set up the metadata file for each protocol before start processing enrollments for the same.

These are common configurations for both Java and .NET RandoNode. The specific details may be little different, that is indicated in the following sub sections.

2.1 Configure \$CTSU_HOME Variable

CTSU_HOME environment variable needs to be setup to use RandoNode Starter Kit. General setup is to make it point to C:/CTSU/Applications/ or D:/CTSU/Applications/ in windows environment. However user can point this variable to other directory if required.

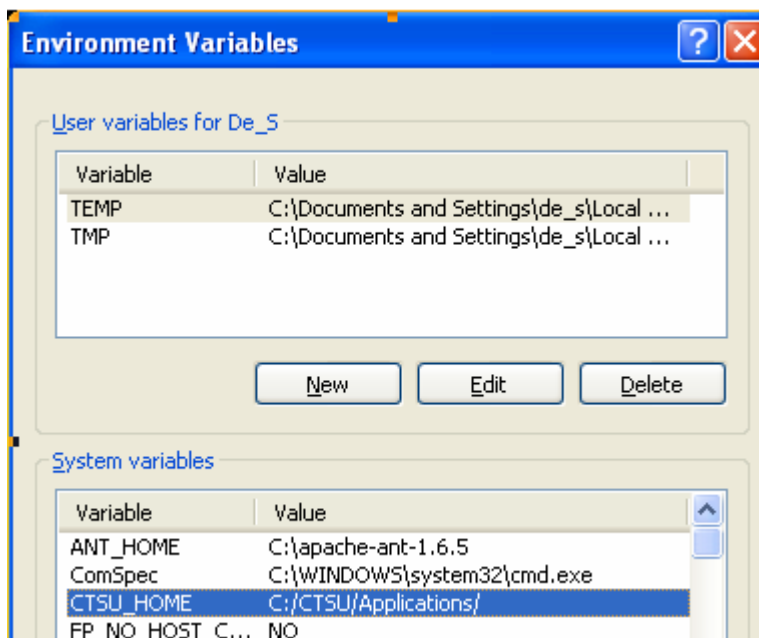


Figure 1 : Setup CTSU_HOME

2.2 How to Setup Demographic Public Id Mapping File

A mapping file named DemographicsPublicIdMapping.xml was introduced in RandoNode version 1.1.

This file contains the mapping of demographic tag to the corresponding CDE public id. Default path of this file is \$CTSU_HOME\RandoNode\config. Users may select another directory to place this file and define the path in configuration file Application.xml. The detail of the configuration file setup is provided in section 4.1. An extract of DemographicsPublicIdMapping.xml file has been provided below.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- <?xml version="1.0"?> -->
- <odmToDemographics>
- <attribute name="lastInitial">
  <publicId>123456</publicId>
</attribute>
- <attribute name="firstInitial">
  <publicId>2658182</publicId>
</attribute>
- <attribute name="middleInitial">
  <publicId>2658163</publicId>
</attribute>
- <attribute name="patientSsn">
  <publicId>780</publicId>
</attribute>
- <attribute name="patientHospitalNbr">
  <publicId>905</publicId>
</attribute>
- <attribute name="ethnicity">
  <publicId>2192217</publicId>
</attribute>
```

Figure 2 : Demography Public Id Mapping

2.3 How to Install Meta Data Files

Copy metadata xml files under \$CTSU_HOME/RandoNode/meta directory for Java RandoNode. \$CTSU_HOME is the environment variable set up in section 2.1. If CTSU_HOME variable is set to C:/CTSU/Applications, metadata files should be placed under C:/CTSU/Applications/RandoNode/meta directory.

For .NET RandoNode, metadata files are installed under C:\CTSU\Applications\RandoNode\Net\meta folder.

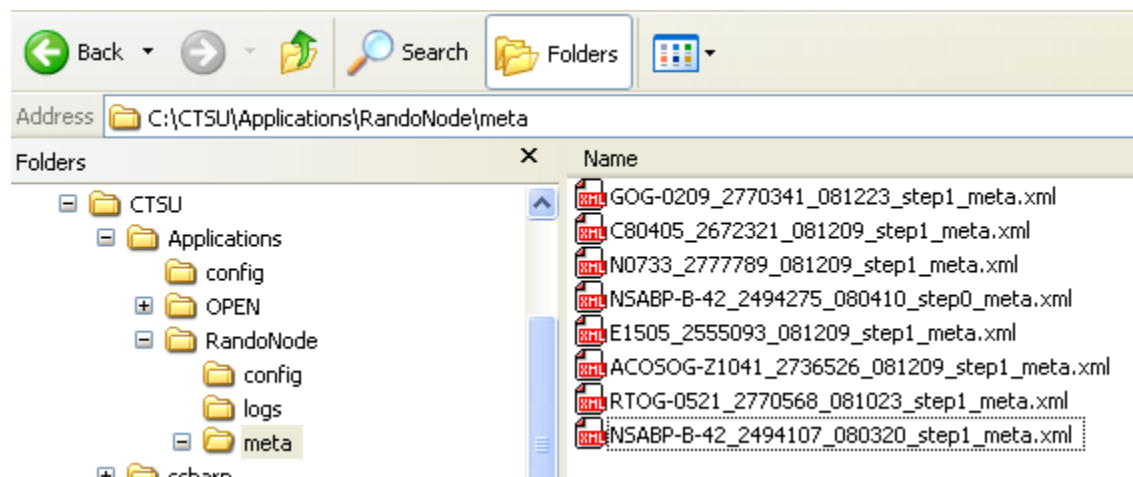


Figure 3 : Meta data files

A new metadata file is generated whenever a new form is downloaded in OPEN and saved, or the existing form definition is changed. LPOs can download the metadata files directly from the OPEN Data Service located at <https://uat-open.ctsu.org/OPENData/services/OPENDataService> . The documents and files related to the OPEN Data Service can be found at https://www.ctsu.org/open/default.asp?fName=open/OPEN_Conference/IT_Conf/java/task3

3 Packages / Namespaces Contained within the RandoNode Starter kit

The following packages (Java) or namespaces (.NET) are contained within RandoNode libraries.

- com.westat.ctsu.open.node [Java and .NET]: This package/namespace contains the RandoNode web service code, the web interface objects and the utility methods to convert between object and xml. This package is provided as the starting point of the web service code.
- com.westat.ctsu.open.node.odm [Java and .NET]: This package/namespace contains the classes that represented the ODM object model and few utility classes.
- com.westat.ctsu.open.node.framework [Java and .NET]: This package/namespace isolates web service code from the randomization business logic. This framework provides a clean separation between the web service input and the business logic.
- com.westat.ctsu.open.node.domain [Java], com.Westat.CTSU.OPEN.Node.Persistence [.NET]: This package/namespace contains the persistent domain objects for working with hibernate. This is optional. If the LPO decides to persist input data for audit trail, or for verification process, then they can refer to this persist package to save data in the database.
- com.westat.ctsu.open.node.persist [Java]: This package is hibernate persistent layer.
- com.westat.ctsu.open.node.example [Java]: This package contains a few examples on how to test the RandoNode.
- RandoNodeTester [.NET]: Similar to the example package above in Java, this folder in .NET RandoNode contains the files which are part of the test application (user Interface) that can be used to invoke the RandoNode instead of using the OPEN application.

4 Step By step Process for Java RandoNode

Java RandoNode starter kit 3.0 is supported in jdk 1.6 and above.

This section describes process of installing RandoNode starter kit 3.0, as well as upgrading an existing RandoNode to starter kit version 3.0. All the bullets below are applicable for RandoNode installation, whereas bullet #1 is not relevant for upgrade.

1. How to Setup RandoNode Configuration File
2. How to install the RandoNode.jar file or the RandoNode Starter Kit code
3. How to build the project and generate RandoNode.war file
4. How to run the sample example files

4.1 How to Setup RandoNode Configuration File

RandoNode uses Application.XML for configuration. This file needs to be setup properly before RandoNode is launched. Application.XML file location depends on the environment variable CTSU_HOME. Application.XML should be placed under \$CTSU_HOME\RandoNode\config directory. This file needs to contain the RandoNode Application node as shown below. LPO needs to modify the GroupRandoNodeApp node value to their corresponding RandoNode<App> class name and GroupRandoNodePackage node value to their specific package. In RandoNode version 1.1, a new tag DemographyMappingDir was introduced. If LPO has Application.XML already setup for previous

RandoNode version, no configuration change is required. Following example snippet depicts the RandoNode configuration for NSABP.

```
<Applications>
  <Application Name="RandoNode" Package="com.westat.ctsu.open.node">
    <Email>ravirajaram@westat.com</Email>
    <GroupRandoNodeApp>RandoNodeNSABP</GroupRandoNodeApp>
    <GroupRandoNodePackage>com.NSABP.ctsu.open.node</GroupRandoNodePackage>
    <PersistData>false</PersistData>
    <LogDir>C:/CTSU/Applications/RandoNode/java/logs</LogDir>
    <MetaDataFileDir>C:/CTSU/Applications/RandoNode/meta</MetaDataFileDir>
    <DemographyMappingDir>C:/CTSU/Applications/RandoNode/config</DemographyMappingDir>
  </Application>
</Applications>
```

4.2 How to Install RandoNode.jar File or RandoNode Starter Kit Code

This section is the core of RandoNode installation/upgrade process. Steps for upgrading from a previous version of RandoNode starter kit to version 3.0 start from here.

4.2.1 Details of RandoNode.jar

The main jar file for RandoNode is RandoNode.jar. This jar contains the classes for the RandoNode framework developed by CTSU. This framework is designed to have a clean separation between the web methods and the LPO's implementation that handles the LPO specific business logic. RandoNode jar contains:

- Classes for the web service entry point. The RandoNode.class contains the interfacing web methods required to interact with OPEN.
- Classes to set up the interfacing objects and partially fill in the response objects returned to OPEN.
- Standard framework classes which can be used across all LPOs.
- Classes to extract data from the metadata file and the clinical data file received from OPEN through the web method, and to link the metadata to the clinical data. Several utility methods have also been provided within these classes for LPOs to retrieve data in various formats. Please refer to https://www.ctsu.org/open/Group_Resources/Randonode/Documents/RandoNode_API.pdf for details of the classes.
- Example classes portray how to persist data in a pre existing database.

4.2.2 Installation/Upgrade Steps

LPOs can use the RandoNode.jar distributed within RandoNode.zip file or the source code provided in the same zip file to integrate the RandoNode starter kit with their business logic. Following two sub sections indicate the steps to install the RandoNode.jar file, and the source code.

LPOs need to keep a backup of their existing development directory before they start merging the new files and follow the steps provided below:

1. Extract RandoNode.zip files in a directory.
2. All .jar files provided with RandoNode.zip lib folder, except RandoNode.jar should be copied under development library (generally called lib) folder.
3. LPO needs to place the latest RandoNode.wsdl and OpenRequestObjects.xml file under RandoNode directory along with the existing files.
4. In case, the existing LPO RandoNode version is lower than 1.1, LPO needs to add the Demographic Public Id Mapping File and update the configuration file. Application.xml needs to contain an entry for the holding directory. The detail is available at section 4.1 and 2.2 of this document.

5. Follow section 4.2.2.1 or 4.2.2.2 to access the RandoNode starter kit version 3.0 class file. They are mutually exclusive, having any one of them is sufficient.

4.2.2.1 Integrate RandoNode.jar file

Delete the existing RandoNode.jar file for RandoNode workspace **lib** directory and place the new file under the same directory.

4.2.2.2 Integrate RandoNode starter kit source code

This is an optional step, needs to be performed only if LPO wants to build RandoNode application from the source code directly, instead of using RandoNode.jar file as described above. LPOs need to perform either of these two steps, not both. Source code contains code for all the Java packages mentioned in section 3.

To install the latest source code in workspace, LPOs should delete (backed up previously) the existing contents of Randonode\src\com\westat folder. Copy the contents of extracted Randonode\src\com\westat folder under the same directory in the development workspace. LPO specific code should be placed under src.com.<appname>.ctsu.open .node package.

4.3 How to Build the Project and Generate RandoNode.war File

The below sections assume that the LPOs use C:/RandoNode as development directory and use Eclipse as the Integrated Development Environment (IDE).

After the latest starter kit code is added to the development directory and all the associated configuration and files are setup using the previous steps, create a new project from Eclipse using the existing files.

Configure the jar files from the RandoNode/lib folder to project build path. Right click on the project, click on Build Path, Configure Build path to see the window provided below. Click on **Add JARs** to add all the JAR files under RandoNode/lib folder to the build path if those JARs are not already added.

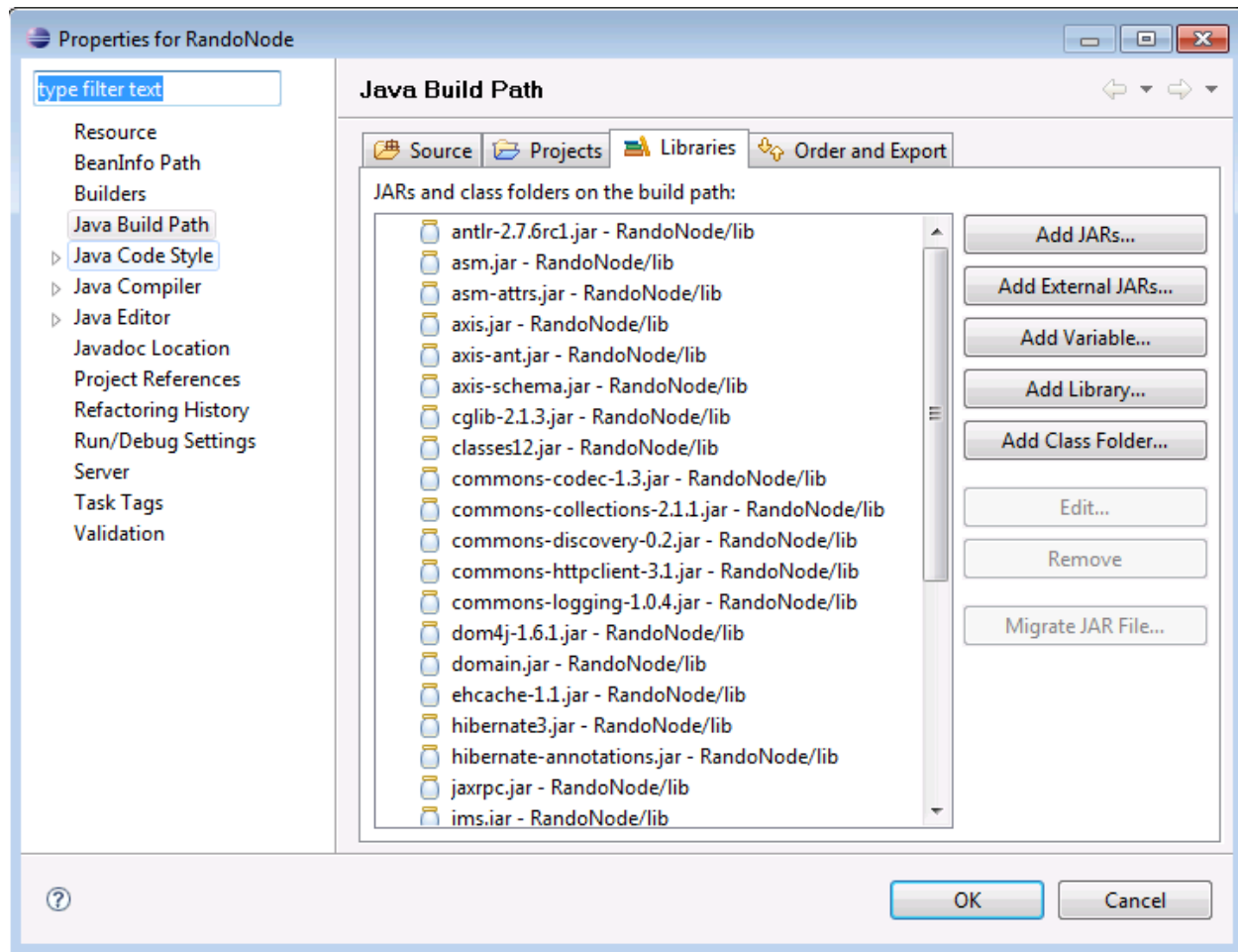


Figure 1: Add Libraries to the Build Path

It is recommended to build the project using build.xml to make sure all the files have been installed properly before the business logic development starts.

Once the project is created, LPOs can start working on the RandoNode business logic development. LPO needs to create src/com/<appname>/ctsu/open/node directory under RandoNode directory if LPO is setting up the RandoNode workspace for the first time. E.g. <appname> is SWOG if SWOG is the LPO, NSABP if NSABP is the LPO etc. Then place LPO specific classes (RandoNodeNSABP.java, RegistrationNSABP.java, RegistrationNSABPB42.java etc.) under this directory similar to previous version. This step is not needed if you are just upgrading your RandoNode.

After the development is complete, run the build.xml file using ant task, RandoNode.war file will be generated under RandoNode/deploy folder for deployment.

4.4 How to Test the Latest RandoNode Installed

RandoNode starter kit provides a few example files under src.com.westat.ctsu.open.node.example package to test RandoNode as a standalone application, without interacting with OPEN portal. Two types of example files have been provided:

- To test RandoNode within development workspace, without deploying RandoNode as web service.
- To test RandoNode as web service.

There are four example files. OpenPortalProxy.java is capable of testing locally all the interfacing

methods of RandoNode. Other three classes are provided for testing all the interfacing methods of RandoNode as web service.

Sample clinical data file is passed as argument to run OpenPortalProxy and RandonodeClient classes as shown below.

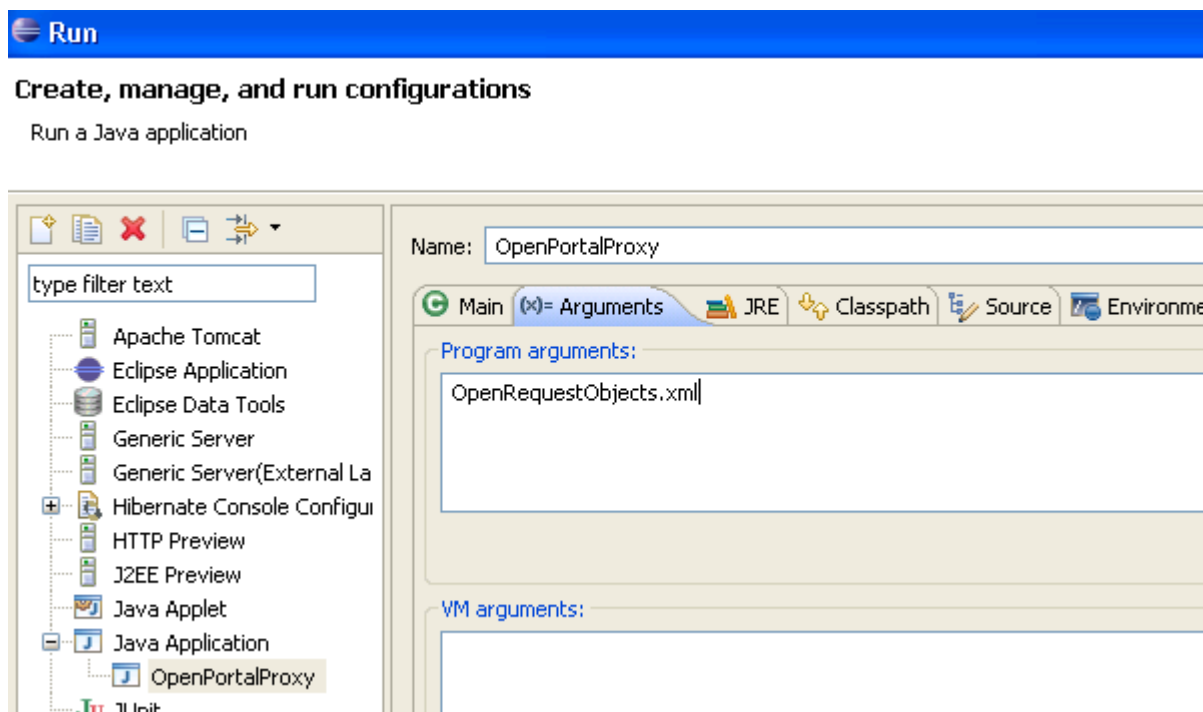


Figure 2: Sending Clinical Data File as an argument when testing the RandoNode code

RandoNode starter kit provides following classes to test RandoNode.

- **OpenPortalProxy.java** – This class act as proxy to OPEN portal. This class reads the sample clinical data from input XML file, converts the request XML to object and invoke RandoNode's interfacing methods. OpenPortalProxy does not invoke RandoNode as web service, can interact with RandoNode locally within development workspace.

This is a very powerful class and has the ability to test all the interfacing methods present in RandoNode 3.0 wsdl file. Please see below the code snippet from the same. Following code snippet provides example for how to test getPatientData and getVersion methods using this class. LPOs can comment out the existing call to getPatient method and replace this code to call any other interfacing method (e.g. doRegister, doCredential etc.).

```

    * Using the objects created from the XML string, call the web service that
    * accepts the objects as input.
    * Please note getPatientData and doCredential methods do not accept odmData
    * as input, but doRegister, doRegisterTest and doValidate methods need
    * odmData as one of the arguments.
    */
    /* RegistrationResponse registResponse = randomnode.doRegister(openRequest,
        openRegistration, odmData); */
    RegistrationResponse registResponse = randomnode.getPatientData(openRequest,
        openRegistration);
    OpenRegistration openreg =registResponse.getOpenRegistration();
    OpenResponse openres =registResponse.getOpenResponse();

    Demography demography = registResponse.getDemography();
    ExistingPatient[] existingPatientArray = registResponse.getExistingPatientArray();
    if(demography != null){
        System.out.println("OpenPortalProxy - First Initial = " + demography.getFirstInitial()
        System.out.println("OpenPortalProxy - Last Initial = " + demography.getLastInitial());
    }
    if(existingPatientArray != null){
        System.out.println("OpenPortalProxy - ExistingPatientArray Size = " + existingPatientA
        ExistingPatient existingPatient = null;
        for(int i=0; i<existingPatientArray.length; i++){
            existingPatient = existingPatientArray[i];
            if(existingPatient != null){
                System.out.println("OpenPortalProxy - PatientId[" + i + "] = " +
                    existingPatient.getPatientId());
                System.out.println("OpenPortalProxy - TreatingInvCtepId[" + i + "] = " +
                    existingPatient.getTreatingInvCtepId());
            }
        }
    }
}
/** Call to Version 1.1 method getVersion() */
System.out.println("OpenPortalProxy - randomnode version = " + randomnode.getVersion());

```

- **RandonodeClient.java** – This class has similar functionality as OpenPortalProxy. Main differences are:
 - This class invokes RandoNode as web service.
 - This class does not test the isAvailable and getVersion methods. Two separate test classes have been discussed below for the same.

Following lines of code within the class needs to be modified before executing the class. First arrow of the following code indicates the web method name. This line needs to be modified if any method other than getPatientData needs to be invoked.

Second arrow indicates the web service address, should be modified to point to actual target web service.

Third arrow indicates adding parameter odmData to request call, e.g. doValidate doRegister methods use this parameter. This line should remain commented for invoking getPatientData and doCredential methods, since these methods do not accept odmData.

The last arrow indicates whether odmData will be passed as input parameter to invoke a method or not. The odmData should not be added to invoke getPatientData and doCredential as discussed above.

```
// String webService = "doRegister";
String webService = "getPatientData";
String groupTNS = "urn:node:open:ctsu:westat:com";
Service doRegisterservice = new Service();
doRegisterservice.setMaintainSession(true);
Call call = (Call)doRegisterservice.createCall();

/**
 * This is the target URL. This URL needs to be modified before invoking the service.
 */
call.setTargetEndpointAddress(new java.net.URL("http://test.service.url/RandoNode"
    + "/services/RandoNode?wsdl"));
call.setUseSOAPAction(true);
call.setEncodingStyle(null);
call.setOperationName(new javax.xml.namespace.QName(groupTNS, webService));
call.addParameter("openRequest", new javax.xml.namespace.QName(groupTNS, "openRequest"));
call.addParameter("openRegistration", new javax.xml.namespace.QName(groupTNS, "openRe
// call.addParameter("odmData", new javax.xml.namespace.QName(groupTNS, "odmData"), Od

OpenRequest.registerTypeMapping(call);
OpenRegistration.registerTypeMapping(call);
OdmData.registerTypeMapping(call);
call.setReturnClass(RegistrationResponse.class);
RegistrationResponse.registerTypeMapping(call);
call.setReturnType(new QName(groupTNS, "RegistrationResponse"));

/**
 * Using the objects created from the XML string, call the web service that
 * accepts the objects as input
 */
/* RegistrationResponse registrationResponse = (RegistrationResponse)call.invoke
    (new Object[] { openRequest, openRegistration, odmData }); */
RegistrationResponse registrationResponse = (RegistrationResponse)call.invoke
    (new Object[] { openRequest, openRegistration});
```

- **IsAvailable.java** – This class is used to test RandoNode isAvailable method as web service. Similar to RandonodeClient.java, EndPointAddress needs to be modified to point to target web service.

```
/* Please change to target RandoNode URL */
```

```
call.setTargetEndpointAddress(new java.net.URL("http://10.61.1.222:" +
    "8081/RandoNode/services/RandoNode"));
```

- **GetVersion.java** – This class was introduced in RandoNode version 1.1 to identify the RandoNode version. GetVersion.java is used to test RandoNode getVersion method as web service. Similar to RandonodeClient.java and IsAvailable.java, EndPointAddress needs to be modified to point to target web service.

5 Step By step Process for .NET RandoNode

This section describes the steps for upgrading .NET RandoNode from older version to RandoNode version 3.0. These steps do not apply for initial .NET RandoNode installation.

Note: Please note that this RandoNode is upgraded to .NET framework version 4.0. Therefore, please make sure you upgrade your existing RandoNode to .NET framework version 4.0 before the migration to this latest RandoNode version.

5.1 How to Upgrade Existing RandoNode

To upgrade the .Net RandoNode to version 3.0, you can follow the steps described below:

1. Back up existing .NET RandoNode source code directory prior to .NET RandoNode version 3 upgrade
2. Replace the following DLLs provided in version 3 distribution
 - a. Framework.dll – RegistrationCore.cs is updated with the new version number.
 - b. Node.dll – OpenRegistration.cs is updated to include the four new data points for the T&UM integration.
 - c. zdk.dll – ZappBlock.cs is updated to include the static string values for the new T&UM operations.
 - d. Persistence.dll – this is needed if this DLL is utilized for persisting the domain objects.
3. Rebuild the RandoNode solution.
4. Develop business logic in LPO specific classes – The doRegister method implemented by the LPO should be updated to include the handling of the new T&UM attributes and the operations. Please refer to the CTSU-RandoNode_API_3.0.pdf document regarding the details of setting of the attributes and the doRegister input/output value definitions for each T&UM operations.

The .NET RandoNode provides an example class that illustrates the T&UM attributes setting based on the CTSU-RandoNode_API_3.0.pdf. This example class is the same as the version 2.0 class except with the added logic to handle T&UM functionalities. The example class is RegistrationSWOG.cs – the doRegister method is modified to illustrate how to set the values of the T&UM attributes. This class is located under RandoNodeStarterKit/RandoNode/App_Code.

5. On your Visual Studio IDE, update the RandoNode web reference for all projects that refers to the RandoNode web reference.

5.2 How to Test the Latest RandoNode

This distribution contains an example clinical data file with its corresponding meta data file. The clinical data file is E1609_191067_Clinical.xml and the meta data file is E1609_3229501_140529_step1_meta.xml. LPOs can use the RandoNodeTester to test the upgraded RandoNode, without involving OPEN. The RandoNodeTester contains RandoNode web reference. Please update the RandoNode web reference on the Visual Studio IDE prior to testing.

Below screenshot shows the output from the RandoNode as displayed on the RandoNode Tester after a successful upgrade. The OpenRegistration object returned from the .NET RandoNode should contain the new T&UM attributes value set either by the RandoNode or from the enrollment form after a successful registration.

```
<caseNotes>test credential v3</caseNotes>
<action>ENROLL</action>
<treatmentAssignmentCode>OTHER</treatmentAssignmentCode>
<treatmentAssignmentDescription>Cisplatin 100mg/m2 IV over 1 hour on Day 1, every 21 days and Taxol 130mg/m2 IV over 3 hours on Day 1, every 21
days</treatmentAssignmentDescription>
<subgroupCode>SG1</subgroupCode>
<diseaseCode>10053650</diseaseCode>
```

Figure 3: Example output received from the RandoNode after a successful upgrade