CTSU

Cancer Trials Support Unit

OPEN RandoNode Request Processing Interface (Updates for RandoNode Version 3.0.0.0)

Revision 05

July 28, 2014

Document Information

Revision Information for the OPEN RandoNode Request Processing Interface (Updates for RandoNode Version 3.0.0.0)

Revision History			
#	**Date**	**By**	**Description**
0a	28-May-2014	Sushmita De	Initial version.
0b	02-Jun-2014	Sushmita De	Updated workflow diagrams.
0c	05-Jun-2014	Mark Stauffer	Updated the document styles and filename to current standards. Replaced the PowerPoint diagrams with Visio diagrams.
0d	06-Jun-2014	Sushmita De	Updated based on the review comments
01	09-Jun-2014	Sushmita De	Updated based on the review comments
02	09-Jun-2014	Kasi Perumal	Review and added TUM data point contents
03	11-Jun-2014	Jayan Nair	Updated overview section to add field details
04	02-Jul-2014	Sushmita De	Updated overview and Example for LPO code changes sections to incorporate LPO feedback
05	28-Jul-2014	Sushmita De	Updated TAD CDE, Disease Code data type and related examples
Last Saved By Sushmita De on 7/28/2014 12:17:00 PM			
File Location: \\westat.com\dfs\CTSU6181\TO2\6181.06_OPEN\Releases\6.1\RandoNode\CTSU-RandoNode_API_3.0.docx			

This document was prepared by:

WESTAT, Cancer Trials Support Unit
1600 Research Boulevard
Rockville, Maryland 20850

Table of Contents

List of Figures

References

#	Document	Location	Description
1.	OPEN RandoNode Request Processing Interface (RandoNode_API)	https://www.ctsu.org/open/Group_Resources/Randonode/Documents/RandoNode_API.pdf	Original RandoNode Request Processing interface document
2.	OPEN RandoNode Request Processing Interface (RandoNode_API_1.1)	https://www.ctsu.org/open/Group_Resources/Randonode/Documents/RandoNode_API_1.1.pdf	RandoNode 1.1 API version. Used mainly for patient population and duplicate patient verification.
3.	OPEN RandoNode Request Processing Interface (RandoNode_API_2.0)	https://www.ctsu.org/open/Group_Resources/Randonode/Documents/RandoNode_API_2.0.pdf	RandoNode 2.0 API version. Used mainly to support the embedded and standalone ancillary studies.
4.	OPEN RandoNode Request Processing Interface (RandoNode_API_2.0.1.0)	https://www.ctsu.org/open/default.asp?fName=open/Group_Resources/Randonode/Documents	RandoNode 2.0.1.0 API version. Used mainly to support the validation of prerequisite data elements.
5.	T&UM Frequently Asked Questions (CTSU-OPEN-T&UM_LPO_FAQ.pdf)	https://www.ctsu.org/open/Group_Resources/Training/Users_Manual/CTSU-OPEN-TUM_LPO_FAQ.pdf	FAQ on Transfer & Update Module

1. Introduction

This document is a supplement to the "OPEN RandoNode Request Processing Interface document" API documents listed in "References" section towards the beginning of this document. The first version of the document described the interface and communication protocols between OPEN portal and the RandoNode, and subsequent versions indicate the version specific design changes. The objective of this document is to describe the changes brought in by RandoNode version 3.0.

This document focuses on the changes required in the LPO RandoNodes to implement a unified approach for identifying and transmitting the additional randomization data points needed for CDUS and SAE integrations. In addition, this document also depicts how to handle different transfer and update requests to support post enrollment data updates.

1.1 Overview

The need for this enhancement on the RandoNode is twofold.

1) Enable a simple and uniform way to collect additional data points needed for SAE and CDUS reporting while reducing the need for the site user entering this data. In the current implementation of SAE data collection, these additional data points are manually entered by the site user. By this enhancement, we can reduce the data entry (by automatically fill in the data from OPEN to Rave) and eliminate the associated data entry errors and data mismatch issues.
2) Support automatic data synchronization between OPEN and RandoNode for post enrollment Credentialing and Demography data updates and patient transfers due to the implementation of Transfer and Update Module (T&UM) in OPEN.

1.1.1 Update for SAE and CDUS reporting

There are a few required data points needed for CDUS and SAE reporting by all LPOs which are not currently collected in a standard manner. Some LPOs collect this in OPEN, some collect in Rave and some others maintain it in home grown systems (Study Manger for Rave (SMR) in the case of COG).

These common data points are listed below. LPOs will need the Disease Code CDE to extract the response value from ODM data. All other CDEs are listed here for informational purpose.

Table 1: Details of the new fields added to the OPEN-RandoNode interface

Data Point	Primary Purpose for Collection	Required?	CDE
Disease Code	• CDUS Abbreviated Reporting • SAE Reporting (converted to Disease Name) • **Note 1:** OPEN will map the Disease Code to Disease Name. Both Disease Code and Disease Name will be pushed to Rave. Disease Name will be used in Rave for SAE reporting. • **Note 2:** LPO can decide to collect this from the user using the demography form or decide to send it back from the RandoNode on a per study basis.	Yes [Please see note 2 on the second column]	2004425, V4.0
Subgroup Code	• Needed only for CDUS complete submission. Added to this update since changing the OPEN-RandoNode interface is an involved process and cannot be done often. • For studies that require only abbreviated CDUS reporting, the RandoNode can return NULL for this value.	Conditionally Required	1925 V2.31
Treatment Assignment Code (TAC)	• SAE Reporting • Complete CDUS reporting Please see the note below regarding blinded and non-blinded studies.	Yes	1967 V4.0
Treatment Assignment Description (TAD)	• SAE Reporting • **Note:** TAD is needed only when the TAC value is OTHER.	Conditionally Required	2002699 V5.0

A valid value for TAC is needed for all studies where CTEP is holding the IND for which the values of TAC will be defined when a protocol is approved.

When CTEP is not holding the IND, then TAC may not be defined for that protocol by CTEP. In those cases, the LPO RandoNode can send “OTHER” through the TAC field when the patient is randomized.

When the TAC field value is OTHER, then the TAD value becomes mandatory and should contain the Treatment Assignment Description. Essentially TAD will tell the NCI reviewer of a safety report the details of the treatment received by the patient.

There is a slight variation in the value returned by the RandoNode depending on if CTEP is holding the IND and the study is blinded or not.

1.1.1.1 Scenario for Studies where CTEP is holding the IND

A TAC value is often assigned when CTEP is holding the IND. The following table provides example values expected for TAC and TAD.

Table 2: TAC and TAD details where CTEP is holding the IND

Example #	Treatment Assignment Code (TAC)	Treatment Assignment Description (TAD)	Comment
1	TAC1	NULL	TAC can be used by the NCI reviewer to find the corresponding TAD hence the TAD will be NULL

1.1.1.2 Scenario for Studies CTEP is not holding the IND

In the case of CTEP is not holding the IND, the TAC and TAD values will be different depending on whether the study is blinded or not.

The following table portrays the scenario of a non-blinded study.

Table 2: TAC and TAD details for non-blinded studies where CTEP is not holding the IND

Example #	Treatment Assignment Code (TAC)	Treatment Assignment Description (TAD)	Blinded?	TAC is Defined?	Comment
1	TAC1	NULL	No	Yes	If TAC is defined for a study, then TAC can be used by the NCI reviewer to find the corresponding TAD hence the TAD will be NULL
2	OTHER	150mg Armodafinil qd (in the morning) x ten weeks	No	No	TAC may not be defined for studies where CTEP is not holding the IND. Since TAC is specified as OTHER, the TAD is needed so that the NCI reviewer can find the treatment that is received by the patient

1.1.1.3 Scenario for Blinded Studies

In the case of a blinded study if CTEP is not holding the IND, the TAC value will be OTHER and the TAC description will be BLINDED.

These specifications are clarified in the below table.

Table 3: TAC and TAD details for blinded studies where CTEP is not holding the IND

Treatment Assignment Code (TAC)	Treatment Assignment Description (TAD)	IND held by CTEP?	Comment
OTHER	BLINDED	No	Treatment Details are not included since this is a blinded study

The final objective of the RandoNode enhancement is to device a uniform way to collect these data points in OPEN across the LPOs so that the downstream processes that depend on these data points can be integrated and automated. This effort is expected to provide the following benefits.

- Uniform data entry across all LPO protocols for these data points which will help the CRAs.
- Easier programming of edit checks which will help the study builders.
- Move us one more step closer to tighter NCTN integration since it will help to build similar processes within and between the protocols managed by NCTNs – SAS extraction, CDUS and SAE Reporting etc. can use a unified logic across all protocols.

1.1.2 Update for T&UM

Transfer & Update Module (T&UM) is an upcoming OPEN feature that will allow sites to electronically update the post enrollment Credentialing and Demography data as well as patient transfer from site A to site B. Once the data update is approved by LPOs and finalized in OPEN, the modified data will be sent to RandoNode for synchronization. The enhanced (version 3.0) RandoNode should be able to retrieve the updated data and persist in the LPO database. If the automatic data synchronization between OPEN and RandoNode fails, OPEN will provide an ability to resend the updated data to RandoNode by push of a button on OPEN screen. Following are the three operations that will be supported for post enrollment updates.

1) Transfer patient from one site to another
2) Update credentialing data
3) Update demographic data

LPOs will have option to subscribe to all three operations, or to some of these based on their needs.

1.2 Purpose

The main purpose is to document the changes needed in RandoNode Starter Kit and provide some examples for the code changes needed in the LPO specific RandoNode classes. This document will serve as guideline for both the Java and .NET RandoNode changes.

1.3 Audience

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

2. Workflow

In the next four subsections, we will review the workflow for four different scenarios as listed below.

1. Capturing of new Elements (TAC, TAD, Subgroup Code and Disease Code)

   This diagram depicts how the new elements can be collected in OPEN. TAC and Subgroup Code are always sent by RandoNode as part of registration response.

   The decision to collect the disease code in OPEN or to populate it from the RandoNode is made by the LPO at the protocol level. If the LPO decide that the disease code will be populated from the RandoNode, then they will disable the disease code field in OPEN using the form setup tab in OPEN.

   TAD is also sent by RandoNode as part of registration response. TAD is mandatory in case the corresponding TAC value is “OTHER”.

2. Process Site Transfer Request

   Site Transfer workflow diagram depicts the overall flow for how the patient gets transferred from one site to another, and the process to automatically synchronize the updated data with RandoNode.

3. Process Credentialing Data Update Request

   Credentialing Data Update workflow diagram depicts the overall flow for updating the credentialing data after the enrollment is complete and the process to automatically synchronize the updated data with RandoNode.

4. Process Demography Data Update Request

   Demography Data Update workflow diagram depicts the overall flow for updating the patient demography data after the enrollment is complete and the process to automatically synchronize the updated data with RandoNode.

2.1 Capture New Elements (TAC, TAD, Subgroup Code and Disease Code)

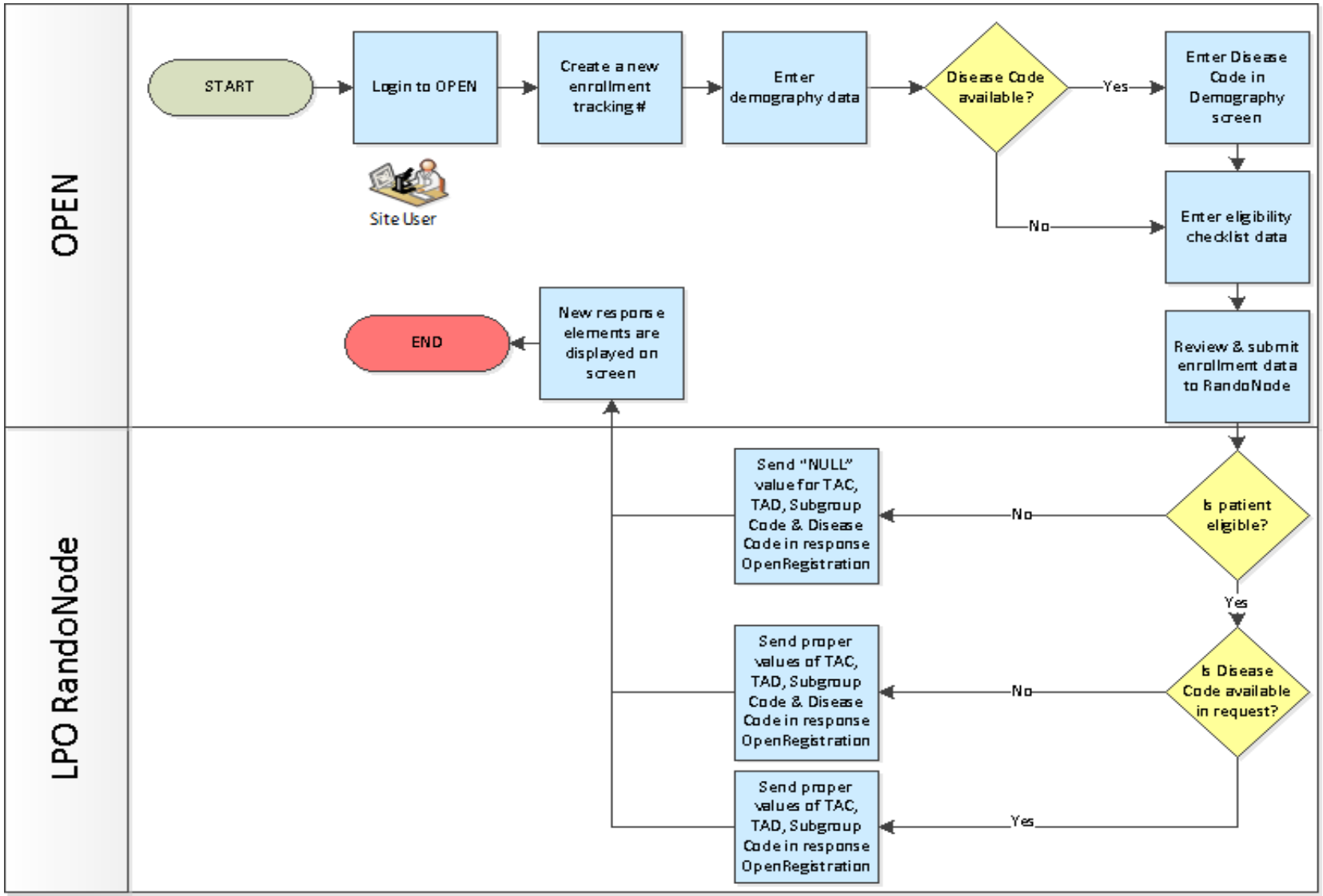

Figure 1: Capturing TAC, TAD, Subgroup Code and Disease Code

2.2 Process Site Transfer Request

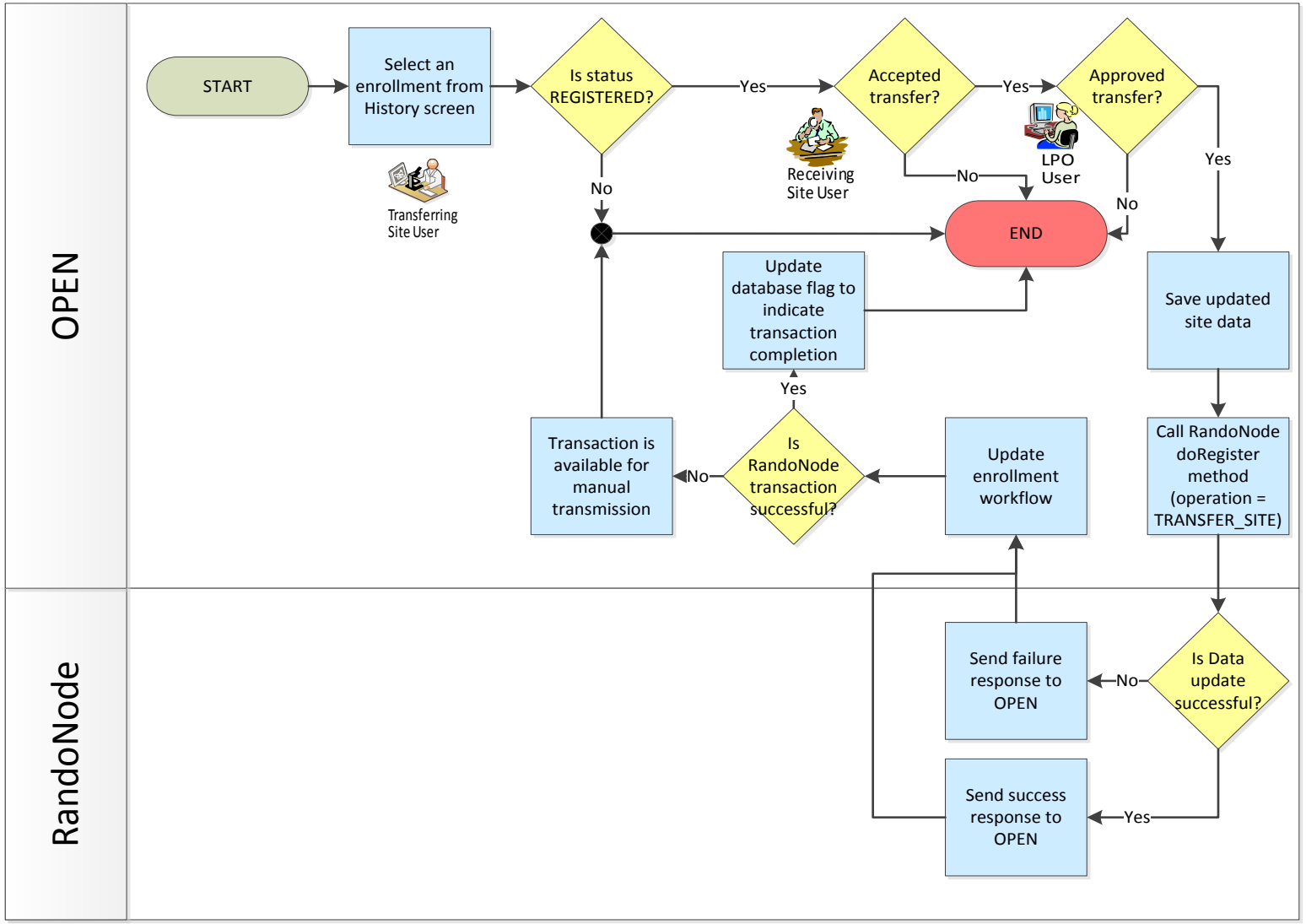

Figure 2: Site Transfer Data Synchronization between OPEN and RandoNode

2.3 Process Credentialing Data Update Request

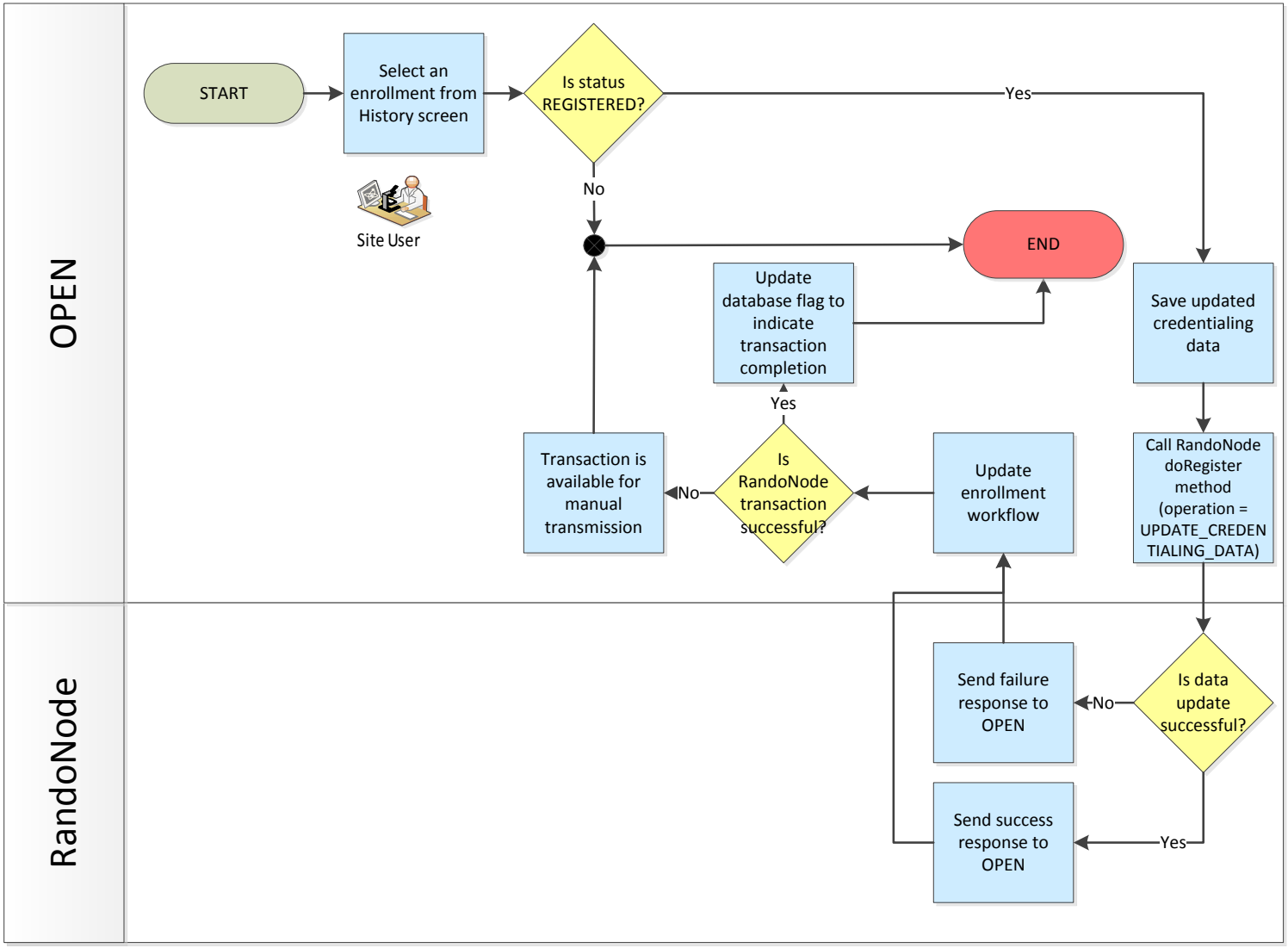

Figure 3: Credentialing Data Synchronization between OPEN and RandoNode

2.4 Process Demography Data Update Request

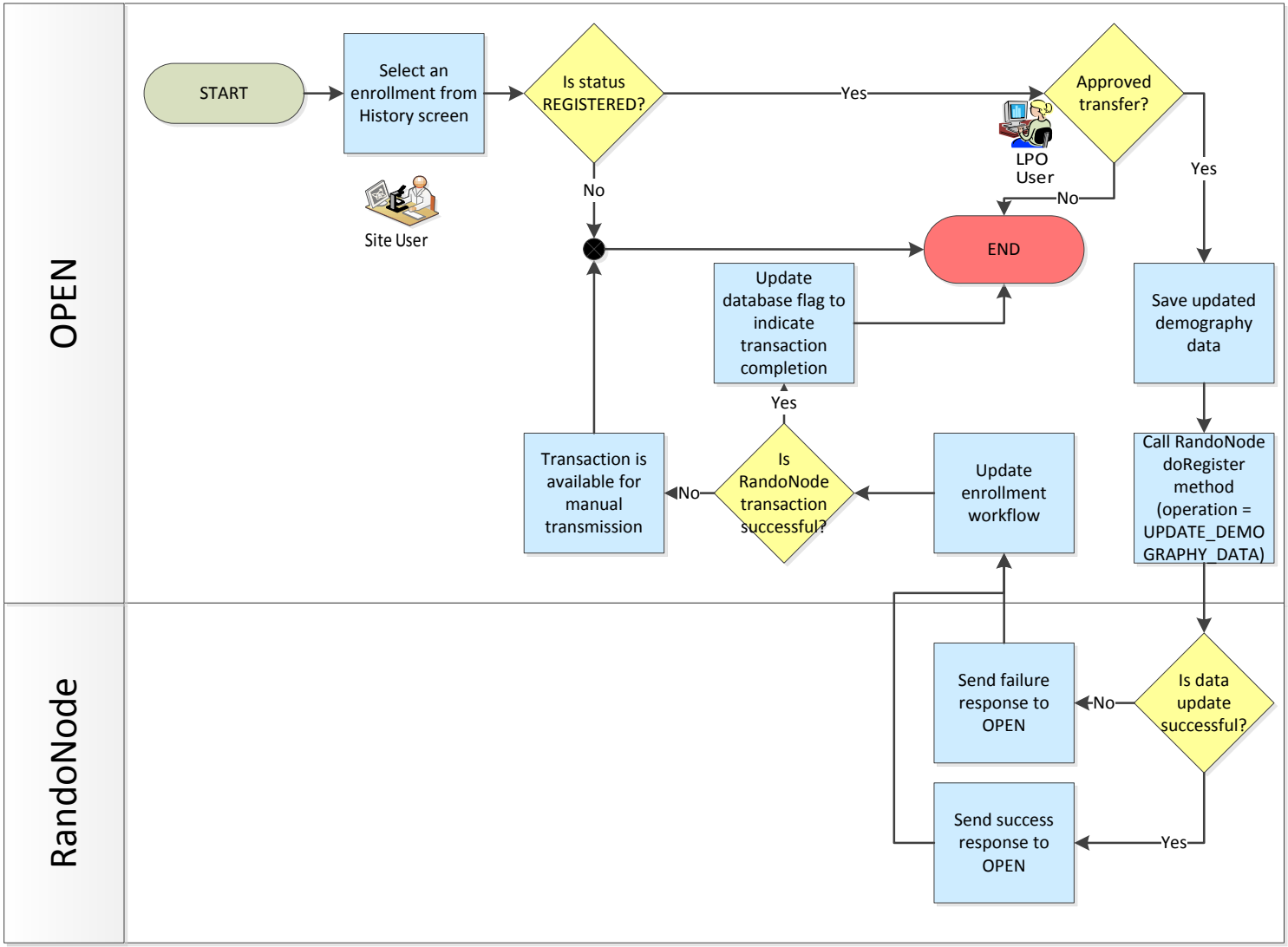

Figure 4: Demography Data Synchronization between OPEN and RandoNode

3. RandoNode Updates

3.1 Design Changes for Starter Kit

3.1.1 Support additional elements in response

OPEN needs to collect Treatment Assignment Code (TAC), Treatment Assignment Description (TAD), Subgroup Code and Disease Code during randomization. RandoNode will send the first three elements as part of the registration response, whereas the fourth element Disease Code is either entered by registrar or received from the RandoNode as a response element if cannot be determined during data entry.

A new question Disease Code is being added in OPEN Demography screen under the module "Standard_NCI_Reporting". For most of the LPOs, site registrar is aware of the patient Disease code during patient enrollment, and will enter the data in OPEN Demography screen. In this case Disease Code value will be sent to RandoNode as part of OPEN request clinical data.

However, for a few LPOs Disease Code value depends on the stratification question responses and determined when the randomization request is processed. For those LPOs Disease Code needs to be sent to OPEN as part of the RandoNode registration response. Disease Code should be populated in response only when the request demography data does not include disease code value.

In such cases, the Disease Code field will be disabled for data entry to the site user, thus implicitly indicates that Disease Code value will be populated as part of registration response object. LPOs need to program their RandoNode to decide whether Disease Code will be received in the request from OPEN or needs to be populated in response. This decision can be made at the study level.

> *Note: If Disease Code value is received in response, OPEN accepts the value only when the enrollment is successful, for all other cases the Disease Code value in response is ignored.*

3.1.1.1 RandoNode.wsdl Enhancements

OpenRegistration object will have four new elements to accommodate the new response elements. They are treatmentAssignmentCode, treatmentAssignmentDescription, subgroupCode and diseaseCode. The treatmentAssignmentCode and subgroupCode are mandatory elements for the RandoNode to fill in. Hence should always be populated by LPOs in the response object.

The treatmentAssignmentDescription and diseaseCode are optional elements. The Data Model diagram below shows the updated elements of the OpenRegistration object.

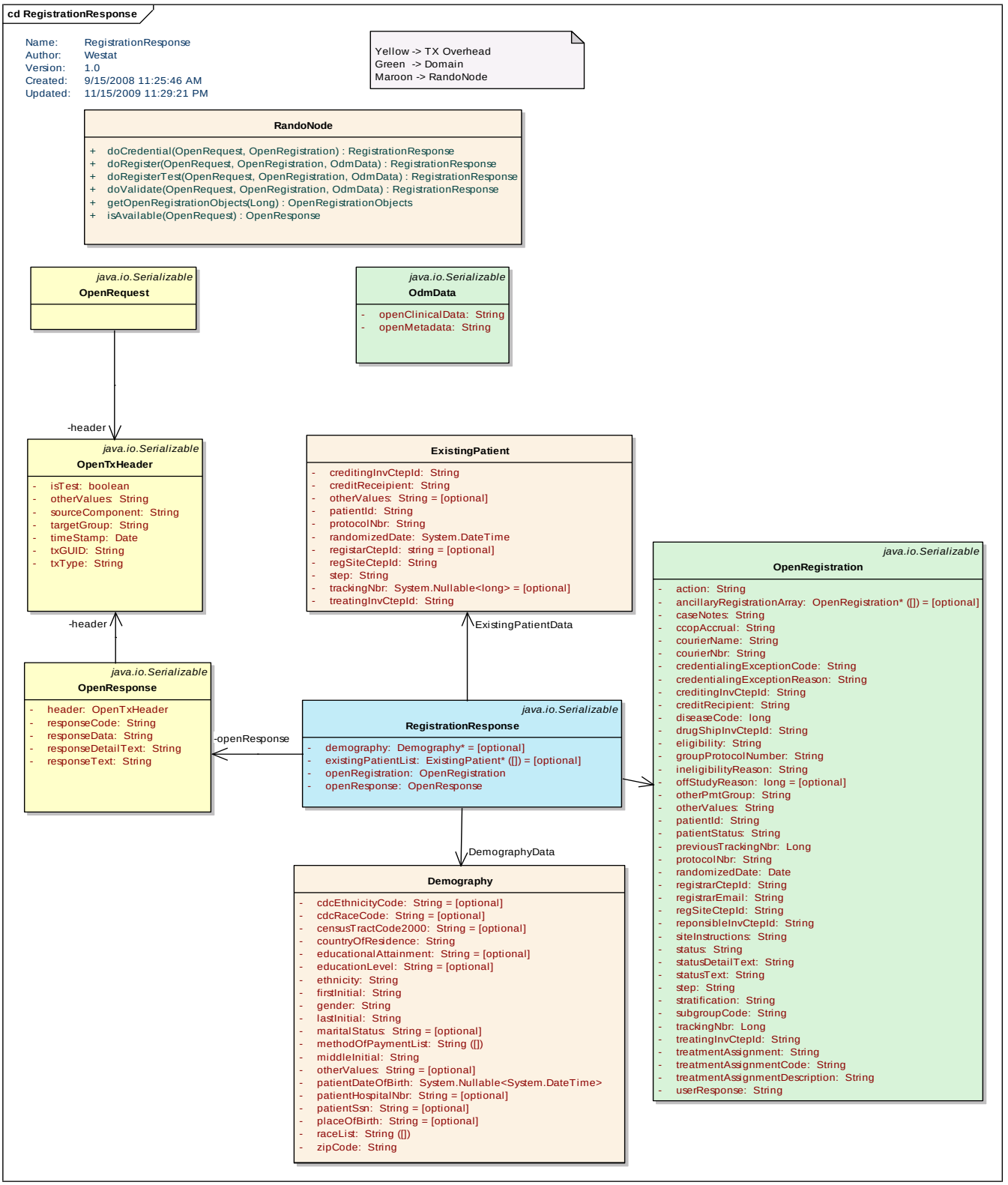

Figure 5: Updated Data Model for RandoNode response object RegistrationResponse

The new elements of the OpenRegistration object are highlighted on the WSDL snippet below.

```
   <complexType name="OpenRegistration">
    <sequence>
     <element name="ccopAccrual" nillable="true" type="xsd:string"/>
     <element name="creditRecipient" nillable="true" type="xsd:string"/>
     <element name="drugShipInvCtepId" nillable="true" type="xsd:string"/>
     <element name="eligibility" nillable="true" type="xsd:string"/>
     <element name="ineligibilityReason" nillable="true" type="xsd:string"/>
     <element name="previousTrackingNbr" nillable="true" type="xsd:long"/>
     <element name="otherPmtGroup" nillable="true" type="xsd:string"/>
     <element name="otherValues" nillable="true" type="xsd:string"/>
     <element name="patientId" nillable="true" type="xsd:string"/>
     <element name="protocolNbr" nillable="true" type="xsd:string"/>
     <element name="randomizedDate" nillable="true" type="xsd:dateTime"/>
     <element name="regSiteCtepId" nillable="true" type="xsd:string"/>
     <element name="registrarCtepId" nillable="true" type="xsd:string"/>
     <element name="registrarEmail" nillable="true" type="xsd:string"/>
     <element name="reponsibleInvCtepId" nillable="true" type="xsd:string"/>
     <element name="siteInstructions" nillable="true" type="xsd:string"/>
     <element name="status" nillable="true" type="xsd:string"/>
     <element name="statusDetailText" nillable="true" type="xsd:string"/>
     <element name="statusText" nillable="true" type="xsd:string"/>
     <element name="step" nillable="true" type="xsd:string"/>
     <element name="stratification" nillable="true" type="xsd:string"/>
     <element name="trackingNbr" nillable="true" type="xsd:long"/>
     <element name="treatingInvCtepId" nillable="true" type="xsd:string"/>
     <element name="treatmentAssignment" nillable="true" type="xsd:string"/>
     <element name="courierName" nillable="true" type="xsd:string"/>
     <element name="courierNbr" nillable="true" type="xsd:string"/>
     <element name="creditingInvCtepId" nillable="true" type="xsd:string"/>
     <element name="userResponse" type="xsd:string"/>
     <element name="patientStatus" type="xsd:string"/>
     <element name="offStudyReason" nillable="true" type="xsd:long"/>
     <element name="groupProtocolNumber" nillable="true" type="xsd:string"/>
     <element name="credentialingExceptionCode" nillable="true" type="xsd:string"/>
     <element name="credentialingExceptionReason" nillable="true" type="xsd:string"/>
     <element name="caseNotes" nillable="true" type="xsd:string"/>
     <element name="action" nillable="true" type="xsd:string"/>
     <element name="ancillaryRegistrationArray" nillable="true"
type="impl:OpenRegistration" minOccurs="0" maxOccurs="unbounded"/>
     <element name="treatmentAssignmentCode" type="xsd:string"/>
     <element name="treatmentAssignmentDescription" type="xsd:string"
nillable="true"/>
     <element name="subgroupCode" type="xsd:string"/>
     <element name="diseaseCode" type="xsd:long" nillable="true"/>
    </sequence>
   </complexType>
```

3.1.1.2 OpenRegistration class

The stub class OpenRegistration will be updated to reflect the WSDL changes mentioned in section 3.1.1.1. OpenRegistration class will be updated to incorporate the new response elements, corresponding getters and setters, as well as the serializer and deserializer.

```
public class OpenRegistration implements java.io.Serializable {public class
OpenRegistration implements java.io.Serializable {
  /**
   * tracking number - the unique number generated by OPEN which represents a
registration.
   */
  private Long trackingNbr;
                    .
                    .
```

```
                    .
                    .
                    .
  // the treatment assignment code
   private String treatmentAssignmentCode;

   // the treatment assignment description
   private  String treatmentAssignmentDescription;

   // the subgroup code
   private String subgroupCode;

   // the disease code
   private Long diseaseCode;               .
                    .
                    .
                    .
                    .
   /**
    * Gets the treatmentAssignmentCode value for this OpenRegistration.
    *
    * @return treatmentAssignmentCode
    */
   public String getTreatmentAssignmentCode() {
           return treatmentAssignmentCode;
   }


   /**
    * Sets the treatmentAssignmentCode value for this OpenRegistration.
    *
    * @param treatmentAssignmentCode
    */
   public void setTreatmentAssignmentCode(String treatmentAssignmentCode) {
           this.treatmentAssignmentCode = treatmentAssignmentCode;
   }

   /**
    * Gets the treatmentAssignmentDescription value for this OpenRegistration.
    *
    * @return treatmentAssignmentDescription
    */
   public String getTreatmentAssignmentDescription() {
           return treatmentAssignmentDescription;
   }


   /**
    * Sets the treatmentAssignmentDescription value for this OpenRegistration.
    *
    * @param treatmentAssignmentDescription
    */
   public void setTreatmentAssignmentDescription(String treatmentAssignmentDescription)
{
           this.treatmentAssignmentDescription = treatmentAssignmentDescription;
   }

   /**
    * Gets the subgroupCode value for this OpenRegistration.
    *
    * @return subgroupCode
    */
   public String getSubgroupCode() {
     return subgroupCode;
   }
```

```
  /**
   * Sets the subgroupCode value for this OpenRegistration.
   *
   * @param subgroupCode
   */
  public void setSubgroupCode(String subgroupCode) {
    this.subgroupCode = subgroupCode;
  }

  /**
   * Gets the diseaseCode value for this OpenRegistration.
   *
   * @return diseaseCode
   */
  public Long getDiseaseCode() {
    return diseaseCode;
  }


  /**
   * Sets the diseaseCode value for this OpenRegistration.
   *
   * @param diseaseCode
   */
  public void setDiseaseCode(Long diseaseCode) {
    this.diseaseCode = diseaseCode;
  }                .
                   .
                   .
                   .
                   .
  elemField = new org.apache.axis.description.ElementDesc();
  elemField.setFieldName("treatmentAssignmentCode");
  elemField.setXmlName(new javax.xml.namespace.QName("urn:node:open:ctsu:westat:com",
"treatmentAssignmentCode"));
  elemField.setXmlType(new
javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema", "string"));
  elemField.setNillable(false);
  typeDesc.addFieldDesc(elemField);
  elemField = new org.apache.axis.description.ElementDesc();
  elemField.setFieldName("treatmentAssignmentDescription");
  elemField.setXmlName(new javax.xml.namespace.QName("urn:node:open:ctsu:westat:com",
"treatmentAssignmentDescription"));
  elemField.setXmlType(new
javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema", "string"));
  elemField.setNillable(true);
  typeDesc.addFieldDesc(elemField);
  elemField = new org.apache.axis.description.ElementDesc();
  elemField.setFieldName("subgroupCode");
  elemField.setXmlName(new javax.xml.namespace.QName("urn:node:open:ctsu:westat:com",
"subgroupCode"));
  elemField.setXmlType(new
javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema", "string"));
  elemField.setNillable(false);
  typeDesc.addFieldDesc(elemField);
  elemField = new org.apache.axis.description.ElementDesc();
  elemField.setFieldName("diseaseCode");
  elemField.setXmlName(new javax.xml.namespace.QName("urn:node:open:ctsu:westat:com",
"diseaseCode"));
  elemField.setXmlType(new
javax.xml.namespace.QName("http://www.w3.org/2001/XMLSchema", "long"));
  elemField.setNillable(true);
  typeDesc.addFieldDesc(elemField);
                   .
                   .
                   .
                   .
```

```
                    .

}
```

3.1.2 Add New Operations for T&UM

Three new constants will be added in ZAppBlock class to represent the T&UM operations. They are

1) **TRANSFER_SITE** – This operation is used to indicate change in enrolling site for patient getting transferred to a different site
2) **UPDATE_CREDENTIALING_DATA** – This operation is used to indicate that some of the credentialing data was updated after the enrollment was completed, e.g. crediting group, courier account number etc.
3) **UPDATE_DEMOGRAPHY_DATA** – This operation is used to indicate that some of the demography data got updated after the enrollment was completed, e.g. last initial, date of birth etc.

3.1.3 Indicate Current Starter Kit Version

The getVersion method of RandoNodeApp class will be updated to indicate the current version of the RandoNode Starter Kit, i.e. 3.0.0.0

3.1.4 Example Client Classes

RandoNode distribution package includes a few example client classes (for Java under the package com.westat.ctsu.open.node.example and for .NET under the namespace com.Westat.CTSU.OPEN.Node.framework) to independently test the RandoNode. The input XML file OpenRequestObjects.xml and the supporting class RegistrationXML.java will be updated to support testing of the new RandoNode Starter Kit version.

3.2 Example for LPO Code Changes

Following sections indicate the changes needed in LPO Registration classes (for Java RegistrationNSABP.java and for .NET RegistrationSWOG.cs). The code provided in these classes is dummy code, example for LPOs how the new changes can be implemented. LPOs need to customize the code based on their specific need.

3.2.1 Example for populating new elements

The doRegister and doRegisterTest method of LPO Registration class will be updated to provide example of populating new elements in response object. Disease Code and TAD are considered as conditional mandatory elements. In case Disease Code value is not available in the request ODM data, RandoNode must be programmed to populate the response OpenRegistration.diseaseCode with appropriate value. If available in the request, OpenRegistration.diseaseCode value can be set to -99999999 in response, which indicates there is no Disease Code value. However if the Disease Code value is populated n RandoNode response, request value will be overwritten in OPEN database.

Using the word “NULL” for strings and -99999999 for numbers is an existing convention for RandoNode to represent an empty data element. If TAD value is not available, LPO should populate OpenRegistration.treatmentAssignmentDescription with the word “NULL”. However, LPO should always populate a valid value for TAD in case TAC value is OTHER. The TAC value will be OTHER whenever a predefined value for TAC is not available, e.g. CTEP does not have IND for the protocol.

Note: Disease Code for a patient is different from one study to another. Hence, Disease Code is not used to retrieve existing patient data or to validate the demography data. For that reason, Disease Code was not included as part of DemographicsPublicIdMapping.xml file that is used to convert the demographic data points from ODM to object elements. So, Disease Code will not be available as part of the existing demography object populated from request ODM data.

LPOs can use the getItemDataByMNameQId(String modName, String oid) method of ClinicalDataUtil class to retrieve the Disease Code from request ODM data. Here, the value to be passed as the modName is ***IG. Standard_NCI_Reporting*** *and oid is* ***ID.2004425*** *for Disease Code.*

The below code provides an example for populating new response elements.

```
  /**
   * This web method contains logic to process registration request
   */
  public boolean doRegister(OpenTxBlock tx,
      RegistrationResponse registrationResponse) {
    boolean successCode = false;
    OpenTxHeader header = registrationResponse.getOpenResponse().getHeader();
    OpenResponse response = registrationResponse.getOpenResponse();
    OpenRegistration registration = registrationResponse.getOpenRegistration();

    try{
                                        .
                                        .
                                        .
                                        .
          if(isValidOperation(operation, "doRegister")){

             /** First check to make sure the meta data is available for this request.
             *  If it is available, proceed, otherwise return error code and message.
             */
              if (checklist.metaDataUtil.isMetaFileAvail()) {
                                        .
                                        .
                                        .
                                        .

               /* Perform randomization based on treating site */
                 String trtAssignment = "";
                 if (checklist.odm.fileOid.indexOf("B-42") >= 0) {
                     trtAssignment = "Blinded";
                 } else {
                     if((Integer.valueOf(patientId) % 2) > 0){
                         trtAssignment = "A";
                      } else{
                         trtAssignment = "B";
                      }
                 }
                 successCode = doRegisterAncillary(registrationResponse,
                                                                    (new
Integer(patientId)).toString(), trtAssignment, false);

                 String treatmentAssignmentCode =
getTreatmentAssignmentCode(checklist);

                 if(successCode){
                  openRegistration.setPatientId(new Integer(patientId).toString());
                  openRegistration.setTreatmentAssignment(trtAssignment);
                 openRegistration.setTreatmentAssignmentCode(treatmentAssignmentCode);
```

```
                    openRegistration.setTreatmentAssignmentDesc(getTreatmentAssignment
                                    Description( checklist, treatmentAssignmentCode));
                    setDiseaseCode(openRegistration);
                    openRegistration.setSubgroupCode(getSubgroupCode(checklist));
                   }else{
                    populateIncompleteResponse(response, "Please cotact CTSU Helpdesk.",
ZAppBlock.PROCESSED, Ancillary study(s) did not meet the enrollment criteria.");
                   }
                  }
               }else{
                  Logger.log(ApplicationInfo.getApplicationName(), "Data Validation
failed.");
               }
               successCode = true;
             }
            }
         }else{
               successCode = doRegisterTest(tx,registrationResponse);
            }
         }
                           .
                           .
                           .
                           .
     }catch(Exception ex){
           Logger.log(ApplicationInfo.getApplicationName(),
ZString.getStackTraceAsString(ex));
           successCode = false;
     }
     return successCode;
   }
```

3.2.2 Support New Operation for Transfer and Update Module

Three new operations are being introduced in RandoNode to support the site transfer and update for post enrollment data. The name and a brief description of each operation are available at section 3.1.2 of this document. The doRegister and doRegisterTest method need to recognize and support these operations appropriately.

The existing isValidOperation method validates the operations supported for a particular method. This method needs to be updated to support the three new operations for doRegister block as shown below.

```
public boolean isValidOperation(String operation, String methodName){
    boolean isValid = false;
    operation = ZString.convertNull(operation);
    methodName = ZString.convertNull(methodName);
    if(operation.length()>0 && methodName.length()>0){
           if(methodName.equalsIgnoreCase("doRegister")){
                  if(operation.equalsIgnoreCase(ZAppBlock.REGISTER_PATIENT) ||
                    operation.equalsIgnoreCase(ZAppBlock.DATA_TRANSFER) ||
                    operation.equalsIgnoreCase(ZAppBlock.MANUAL_REGISTRATION) ||
                    operation.equalsIgnoreCase(ZAppBlock.PROCESS_ANCILLARY_STUDY) ||
                    operation.equalsIgnoreCase(ZAppBlock.TRANSFER_SITE) ||
                   operation.equalsIgnoreCase(ZAppBlock.UPDATE_CREDENTIALING_DATA) ||
                    operation.equalsIgnoreCase(ZAppBlock.UPDATE_DEMOGRAPHY_DATA)){
                         isValid = true;
                  }
           }else if(methodName.equalsIgnoreCase("doValidate")){
                                .
                                .
                                .
                                .
    }
```

```
    return isValid;
  }
```

LPOs need to develop code to persist the modified enrollment data in the database for all three operations. The first step for LPOs will be to verify that the corresponding enrollment status indicates patient is already registered in their system, since these operations are supported only for post enrollment. If a registered enrollment is found, LPO needs to update the relevant field data in their database.

The request data will not contain any flags to indicate the updated data points. Following tables indicate the elements LPOs should be looking for each update.

3.2.2.1 Details of the Fields Related to Patient Transfer

OPEN Field	Mapped Request Data Field	Parent Object	Received Data Type	Comments
Receiving Institution CTEP ID	regSiteCtepId	OpenRegistration	String	The CTEP ID of the site receiving the patient
Persons Associated with the Enrollment (e.g. Treating Investigator, Site Registrar, Contact person for Sample Submission etc.)	treatingInvCtepId, registrarCtepId of OpenRegistration object. AdminData > User	OpenRegistration OdmData	String	The CTEP ID of the treating investigator and the registrar can be found under the OpenRegistration object. However more details of these users and all other registration associates can be found under the AdminData, User node of OdmData. There is no change in the request data format, only the updated values will be populated during this operation
Network Group Credit	creditRecipient	OpenRegistration	String	
Credit Investigator	creditingInvCtepId, AdminData > User	OpenRegistration	String	The CTEP ID of the crediting investigator can be found under the OpenRegistration object. However more details of this user can be found under the AdminData, User node of OdmData.
Express Courier Account Name	courierName	OpenRegistration	String	
Express Courier Account Number	courierNbr	OpenRegistration	String	

3.2.2.2 Details of the Fields Related to Credentialing Data Update

OPEN Field	Mapped Request Data Field	Parent Object	Received Data Type	Comments
Persons Associated with the Enrollment (e.g. Treating Investigator, Site Registrar, Contact person for Sample Submission etc.)	treatingInvCtepId, registrarCtepId of OpenRegistration object. AdminData > User	OpenRegistration OdmData	String	The CTEP ID of the treating investigator and the registrar can be found under the OpenRegistration object. However more details of these users and all other registration associates can be found under the AdminData, User node of OdmData. There is no change in the request data format, only the updated values will be populated during this operation
Network Group Credit	creditRecipient	OpenRegistration	String	
Credit Investigator	creditingInvCtepId, AdminData > User	OpenRegistration	String	The CTEP ID of the crediting investigator can be found under the OpenRegistration object. However more details of this user can be found under the AdminData, User node of OdmData.
Express Courier Account Name	courierName	OpenRegistration	String	
Express Courier Account Number	courierNbr	OpenRegistration	String	

3.2.2.3 Details of the Fields Related to Demographic Data Update

All the following fields can be found as ItemData of the ODM clinical data sent to RandoNode. The ClinicalDataUtil class of Existing RandoNode contains a utility method named getStandardDemographyObject. This method converts the demography data from ODM format to an object. LPO can use the converted demography object to retrieve these field values.

OPEN Field	Mapped CDE Public ID	Parent Object	Received Data Type	Comments
Last Initial	2658183	OdmData	String	TUM Configurable
First Initial	2658182	OdmData	String	TUM Configurable
Middle Initial	2658163	OdmData	String	TUM Configurable
Patient SSN	780	OdmData	String	TUM Configurable – This field is rarely used. LPO can decide not to use it at all
Patient Hospital No.	905	OdmData	String	TUM Configurable
Patient's Date of Birth	793	OdmData	String	TUM Configurable
Ethnicity	2192217	OdmData	String	TUM Configurable
Gender of a Person	2200604	OdmData	String	TUM Configurable
Country of Residence	315	OdmData	String	TUM Configurable
Zip Code	2179606	OdmData	String	TUM Configurable
Race	2192199	OdmData	String	TUM Configurable
Method of Payment	2003309	OdmData	String	TUM Configurable

3.3 Other changes

1) All legacy OPEN studies will be linked to the new Demography form. This will result in a new metadata file. LPOs will need to use this new metadata files for all their studies. CTSU will support the LPOs to upgrade the legacy studies with the new form
2) LPO needs to educate sites about the new Disease Code field response to be entered in the OPEN Standard Demography form

4. How to Install/Upgrade to RandoNode 3.0.0.0 Build

The readme.doc provided along with the RandoNode distribution package indicates the detailed step to install or upgrade the RandoNode Starter Kit.