Cancer Trials Support Unit

CTSU – A Service of the National Cancer Institute

OPEN RandoNode Request Processing Interface

Revision 04

20 February 2009

Document Information

Revision Information for the **Cancer Trials Support Unit OPEN RandoNode Request Processing Interface**
Document No**. CTSU/DOC## Rev. 02**

Revision History			
#	**Date**	**By**	**Description**
01	18-JUL-2002	Rajaram R	Intitial version
02	14_sep_2008	Rajaram R	Updated to make the architecture current as of Sep 2008
03	26_jan_2009	De S	Updated OpenRequest operation types.
04	20_feb_2009	De S	Updtaed Site Registrar scenarios.
Last Saved By **Sushmita De** on **2/20/2009 5:08 PM**			
File Location: **\\rk27\vol2703\CTSU8339\Tasks\8339_20_OPEN\03_Construction\RandoNode\API_Docs\RandoNode_API.doc**			

This document was prepared for:

Template Help:

Use the **File→Properties** option to update footer and key values in document.

This document was prepared by:

WESTAT
1650 Research Boulevard
Rockville, Maryland 20850 Phone: (301) 251-1500

Ravi Rajaram 301 294 2045

OPEN RandoNode Request Processing Interface

Introduction

Overview

OPEN is a web based system that will enable sites to enroll patients in Cooperative Group and other research network protocols. The site users enter the credentialing and Eligibility check list data through the OPEN Portal. On submission, a request is made to the lead Group's RandoNode web service to perform the registration. The RandoNode will process the data from OPEN in order to determine the eligibility of the patient. After processing, the Group's RandoNode will send a response to OPEN system that includes the patient ID and treatment arm etc., if the patient is eligible and other information that is required for performing these enrollments.

Interface Data

The transferred objects include data on enrolling site and investigator as well as the eligibility checklist form data in an XML format (embedded in one of the request classes) that follows the specifications of the Clinical Data Interchanges Consortium's (CDISC) Operational Data Model (ODM) version 1.3.0. The details of the ODM version 1.3 are available at CDISC 1.3 spec.

Purpose of Document

The purpose of this document is to describe the interface classes between RandoNode and Open portal and the processing of the request objects that are sent to the RandoNode as a result of a patient registration through the OPEN Portal.

It is important to understand that this document really describes **two different interfaces**:

1. The web service interface in the form of methods and classes that are exchanged. This interface gets the data to and from the RandoNode. It does not include how to process it once it is at the RandoNode.
2. The API to access data from the clinical data object that contains the eligibility checklist data. This data is transmitted in XML. The ODM API provided can be used as an object-oriented mechanism to access the data.

The use of the web service interface is **required**. The use of the ODM API is not required, but is highly recommended so that your RandoNode can be insulated from changes. A RandoNode can be implemented to utilize the XML directly. This document also describes the CDISC XML format to some degree, however, the CDISC 1.3 spec can be used for this purpose.

This document also describes a staging database schema to which the clinical data can be extracted.

Request API

Overview

To process a request from the OPEN System, each Group's RandoNode must be configured to receive and recognize the OPEN request objects.

The requests that are transmitted from the OPEN Portal to the Randomization Node for every registration will include three parameter objects: **openRequest**, **openRegistration** and, if applicable, **odmData**.

The OpenRequest consists of the transactional data such as transaction GUID, transaction time stamp etc...OpenRegistration contains registration related data such as study name, enrolling site, investigator, enrolling CRA, group receiving the accrual credit enrollment/randomization date etc... The OpenRegistration has attributes for the group to fill out to send back to OPEN portal. odmData holds the clinical data and form meta data and as per CDISC ODM format. Examples of each class/object, their attributes are included below.

Other methods that are defined that are not registration specific may include simply the openRequest object, or other simple data types, depending on their purpose.

RandoNode- Methods (Required)

Overview

In order for Open Portal to communicate with the Group RandoNodes, a defined interface is required. **Each Group RandoNode must implement the methods described below**. The RandoNode methods to be implemented by the Cooperative Groups are defined with the following WSDL:

http://www.ctsu.org/open/Randonode/RandoNode.wsdl

The approach to the design of the method signatures was to avoid implementing special classes for passing data to the methods. The methods will make use of domain classes where possible, and include general purpose request and transaction classes to pass information to the methods. The methods typically accept an OpenRequest object, plus one or more domain objects as parameters.

Because a method can only return a single value or object, special class(es) are required to return data to the caller. A specific response class (e.g., RegistrationResponse) will include overhead objects, as well as one or more domain objects that were operated on by the RandoNode (e.g., OpenRegistration). Details of each specific response class are described below.

otherValues

As you review the details of the classes, you will notice a common attribute named **otherValues**. This attribute is included as a way to include additional values in a **named-value** pair configuration in an XML format without the need to update the signature of the API. Over time, the class definitions used within the interface will change and the system will accommodate the transitioning of one version to the next. However, the use of this additional mechanism can minimize the number of versions of WSDL that are released.

Tretament of Nulls

There maybe cases where there is no value available for the attributes in the OpenRequest, or OpenRegistration object. If there is no value for those attributes, OPEN is going to fill in with NULL for string data type attribute, and to fill in -99 for number data type attribute.

Required Methods to Implement

The following table summarizes the RandoNode method API that must be supported by each Group RandoNode. The description below includes the required input parameters and return values. Note that most of the parameters are classes that are described below.

Method	Parameters		Description
	Input	Output	
isAvailable()	OpenRequest	OpenResponse	The isAvailable is intended to find out if the group RandoNode services are available for accepting requests. If the group RandoNode service is ready, the node should return the OpenResponse.responseCode as **READY**. The value of **NOT-READY** should be returned if the RandoNode is not able to process requests in the event of system maintenance or other causes. If no response is received at all, OPEN will assume **NOT-READY**.
doCredential	OpenRequest OpenRegistration	RegistrationResponse	This method should perform any special site, investigator and CRA information validation. Please note that this is not required in most cases as OPEN validates this information against RSS. This is required when the accrual is credited to the lead group and the lead group is not maintaining the roles data in RSS. The OpenRegistration class contains the registration related data such as protocol, site, investigator, credited Group etc. The doCredential does not need the eligibility checklist data (ODM object). The OpenRegistration.status should be set to SUCCESS if the registering user has enrollment privileges in the lead group database, and the associated personnel have permission to participate in the study. The OpenRegistration.status should be set to FAILURE if the registering user does not have enrollment privileges in lead group database. The OpenRegistration.statusText should contain failure explanation. This message should aid the registrar in fixing the problem and re-initiate the registration. The OpenResponse.responseCode should be set to • PROCESSED if the request is performed successfully • EXCEPTION if the request cannot be completed. The OpenResponse.reponseText should contain a brief reason of why the operation cannot be performed. The OpenResponse.responseDetailText is available for filling in detail message.
doValidate	OpenRequest OpenRegistration OdmData	RegistrationResponse	This is used to validate the ClinicalData (ODM) before invoking the doRegister method. When this method is invoked, groups are expected to validate the data only and do not perform any registration. RegistrationResponse is the output from the RandoNode. The OpenRegistration.status should be set to • SUCCESS if the registration data passes the group's validation check. • FAILURE if the data contains validation error. The OpenRegistration.statusDetailText should contain the validation failure detail. The OpenResponse.responseCode should be set to

Method	Parameters		Description
	Input	Output	
			• PROCESSED if the request is processed without any operational error. • EXCEPTION if the request is not completed due to any operational error such as database is down.
doRegister	OpenRequest OpenRegistration OdmData	RegistrationResponse	The OpenRegistration will contain only the registration related data such as protocol, site, investigator, credited Group etc. The OdmData object will contain the EC data. RegistrationResponse is the output from the RandoNode. The OpenRegistration.status should be set to • SUCCESS if the registration request is completed. • FAILURE if the registration request is completed, but the patient is not eligible for the study. The OpenRegistration.patientId, and OpenRegistration.treatmentAssignment should be filled in by the group if the patient is eligible. The OpenRegistration.eligibility should be set whether the patient is eligible or not. The OpenRegistratoin. ineligibilityReason should contain the reason if the patient is not eligible to be registered into the study. The lead group can use the OpenRegistration. siteInstructions to communicate any additional information for the registering site. The OpenResponse.responseCode should be either PROCESSED or EXCEPTION depending on whether the request is processed without any operational error.
doRegisterTest	OpenRequest OpenRegistration OdmData	RegistrationResponse	This is the registration request for testing purposes. This method will be called only with test data during testing by someone using the OPEN portal.

doRegister Example

Once you have created the web service on your side using the WSDL, the API methods will be empty and you will have to implement each one.

At this point, however, you have the data in easy to use objects, which are described below. It is at this point that the ODM API (described below) to access the data in the eligibility checklist can be utilized, or you can process the XML "manually."

For a registration request (doRegistration), your logic will essentially create a registration record in your own patient registration system based on the OpenRegistration data and the eligibility checklist, and fill in some values in the OpenRegistration object to be returned. To return the OpenRegistration object, you will construct a RegistrationResponse object and place the OpenRegistration, and fill in some response values (e.g., PROCESSED, etc.) The response object is returned to the OPEN Portal for completion of the registration. The RegistrationResponse object contains 2 objects, OpenResponse and OpenRegistration. You have to set the OpenResponse.header object to the OpenRequest.header object. The OpenResponse.header object has to be filled in for all the web methods provided by RandoNode.

Interface Classes

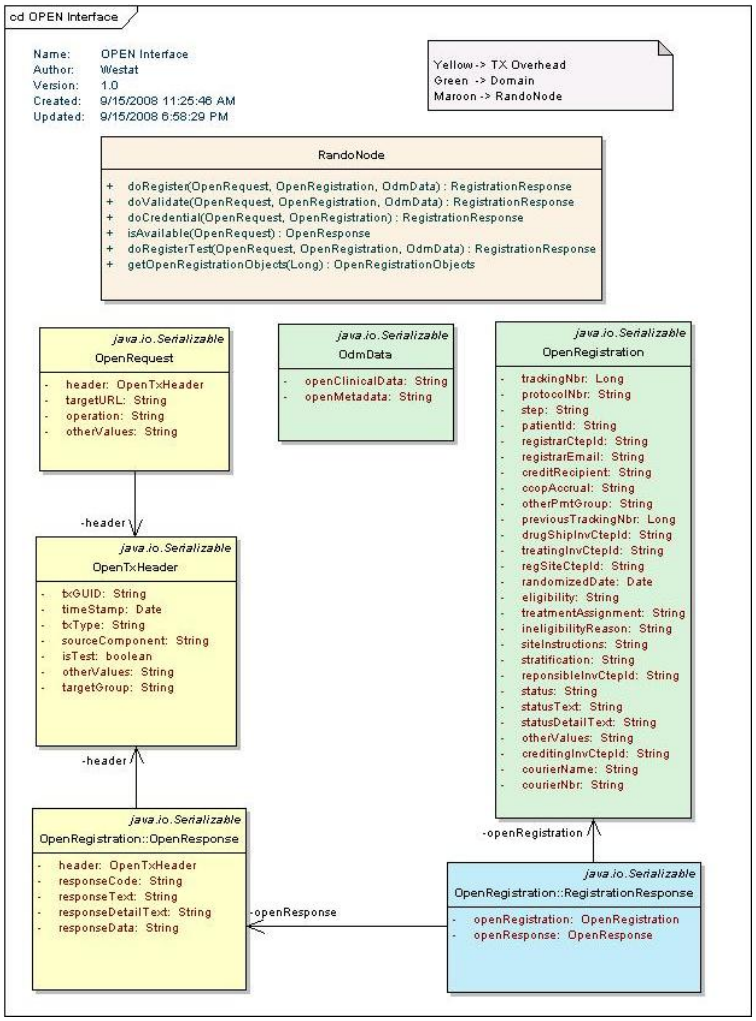
Overview

Currently there are six classes that comprise the interface between OpenPortal and RandoNode. These classes are:

- OpenRegistration
- OdmData
- OpenTxHeader
- OpenRequest
- OpenResponse
- RegistrationResponse

The class diagram for these six classes is shown in the following diagram. For the back ground of class diagrams and the naming convensions, see the section ODM API.

Figure 1 Interface Class Diagram



The description of the classes and the attributes are given in the table below. The organization and

purpose of various classes are explained in the following section.

Domain Classes

Overview

The domain classes used in the interface carry the real application data and are used to communicate this data between the OPEN Portal and the Group RandoNodes. These classes exclude the transaction/request overhead data. The transaction/request overhead classes are described in the next section and are used to augment or "wrap" the domain classes and provide request context, tracking, and response information (including exceptions and other anomalies).

It is expected that the domain objects that are passed to the nodes are updated (e.g., with patient ids, etc.) and returned to the Portal.

In the initial implementation, only two domain classes are used in the API. These two are the OpenRegistration and the OdmData, and are described below.

class OpenRegistration

The **OpenRegistration** class contains registration related data such as study name, enrolling site, investigator, enrolling CRA, group receiving the accrual credit enrollment/randomization date etc... The OpenRegistration has attributes the group fills out to send back to OPEN portal.

Attribute	dataType	Example data	Description
trackingNbr	number	33413	This is the unique tracking number given by the OPEN portal application to identify a patient registration uniquely. The tracking number for different steps for the same patient will be different.
protocolNbr	v35	NSABP-B-42 S0777 E1505	The PIO protocol number for this registration request.
step	v5	1, 2	The protocol step for this request.
regSiteCtepId	v5	MN024 MD017 11027 (Canadian)	The enrolling site. This is the site where the patient is enrolled at.
reponsibleInvCtepId	v7	21961, 18186	The ctepid of the investigator who is responsible for the patient for accrual credit, payment and auditing purposes. This will be removed in future versions. This is equal to the crediting investigatot ctep id.
treatingInvCtepId	v7	21961, 18186	The ctepid of the treating investigator who is treating the patient.
registrarCtepId	v7	502230, 500995	The registrar's CTEP Id. Registrar is the person doing the registration in OPEN portal.
registrarEmail	v240	abc@yahoo.com	The registrar's Email account.
randomizedDate	date	2008-04-15T19:31:26.453Z	This is the date the patient is registered/randomized in to the group registration system. CTSU will prefill this field with the date the site is submitting the registration request. If for any reason, the patient is not randomized on the same date, the prefilled date will be updated upon SUCCESS registration status from group.
creditRecipient	v20	CALGB, ECOG, NSABP	The cooperative group that receives the accrual credit.
drugShipInvCtepId	v7	21961, 18186	The CTEPID of the investigator to whom the study drug will be shipped. This may not be collected for every protocol. This is collected in OPEN portal based on the protocol set up in RSS.

Attribute	dataType	Example data	Description
previousTrackingNbr	number	31412	This is the tracking number corresponding to the previous step for the same patient. Only applicable for protocols with multistep registration.
ccopAccrual	v3	YES, NO	This is a flag that indicates whether the enrolling site is a CCOP site or not. The values are YES or NO
otherPmtGroup	v20	CALGB, NSABP,SWOG	This is not used at this point. It is here for handling patient transfers. To be used by OPEN portal in future.
eligibility	v10	ELIGIBLE, INELIGIBLE, INCOMPLETE	This should be filled in by the lead group after the determination of the patient eligibility. The valid values are • ELIGIBLE – the patient is qualify and registered to enroll in the study. • INELIGIBLE – the patient is not qualified to enroll in the study. This registration should not allow submitting again. • INCOMPLETE – the data contains validation error, and the groupintends to allow the registrar to correct the error and resubmit the same registration.
ineligibilityReason	v4000	Patient's T score is out of range	The reason for patient's ineligibility. This should be filled in by the lead group if the group determines that the patient is ineligible.
patientId	v20	826599998, 52393, USXU0156	patient id, if available. This is to be normally filed by the lead cooperative group. In certain instances, (post registration) OPEN portal will prefill this while sending the request to Group.
treatmentAssignment	v10	A, B, BLINDED	The treatment arm assigned to the patient by the lead cooperative group. To be filled in by the lead group. Required only if the patient is eligible.
siteInstructions	string (longtext)	Please send the lab sample to the following address. 123, Main Street, Tin Town, MD-20850	This should be filled in by the lead group. This is a general use place holder. The group can provide instructions to the site regarding this registration. The contents will be displayed to the registrars in the OPEN portal's registration confirmation page.
status	v32	SUCCESS FAILURE PENDING-GROUP	The status of the registration request. The valid values are SUCCESS, FAILURE, PENDING-GROUP. As long as the data is processed successfully and the patient is determined to be eligible or not eligible, the status will be Success. If the data fails the validation checks, then groups can return FAILURE. If due to some reason, the data patient eligibility or the requested operation such as credentialing, validation can not be determined at this time, PENDING-GROUP can be used. A good example for use of PENDING-GROUP is the case where the Group does not have the metadata file as indicated within the OdmData. Group needs time to download the metaf data file and complete the set up.
statusText	v500	No active IRB approval record for this institution in our database.	Short description of the status which can be displayed to the site on the OPEN portal screen. No status text is required in case of SUCCESS. This is mandatory in case of FAILURE and PENDING-GROUP. This needs to be filled by the Group. .
statusDetailText	string (longtext)	The last IRB approval transaction received was on 12/1/2006. the approval has since expired.	This should be filled in by the lead group to communicate in case of FAILURE or PENDING-GROUP. This is a long version of the status text. The information is for CTSU staff and will not be communicated to the site. This is optionally filled by the group.
stratification	v15	Exes.+Trip, Trip	To be filled in by the lead group. This is Optional.

Attribute	dataType	Example data	Description
otherValues	string (longtext)	XML	This is included to provide a mechanism to pass additional parameters without having to modify the class definition or API. It will be named-value pairs in XML. APIs to access this data easily will be provided.
courierName	V40	AIRBORNE EXPRESS DHL FEDERAL EXPRESS PUROLATOR UPS TNT	This is the courier company. It is patient specific attribute. This is an express courier account name at the time of patient registration in order to expedite the shipment of the initial clinical supplies to the registering clinical site
courierNbr	V20	1234-5678-9	This is the account number associated with the cexpress courier account. The express courier account name and number are associated ONLY with the initial e-order, which immediately follows the patient registration / randomization. Subsequent orders will be sent via the normal shipment mechanism
creditingInvCtepId	V7	21961, 18186	This is the same as responsibleInvCtepId. We will remove the duplicated attribute in the future.

class OdmData

The **OdmData** class contains clinical data collected such as responses to the eligibility checklist questions. This class includes a reference to the openMetaData. Normally, this will be empty when sent in a request as the meta data files will be provided to the groups as part of the configuration setup and the OPEN ODM API will populate this. The details of OPEN ODM API are available in the later sections of this document.

In OdmData oid has been used for FormData, ItemGroupData, ItemData etc. Oid is the Id constructed by OPEN for an element. It is constructed generally using the CDE public Id or long name. For Forms, the format is the word “FORM.” and the form public ID, e.g. FORM.2736526. For ItemGroupOID, it is the word "IG." and the Item Group long name, e.g. IG.Race. For Items, ItemOID is the word "ID.” and the item CDE public Id in general, e.g. ID.791. However there is a special case where the items with same CDE public Id can recur within the itemgroup. In that case order number is also appended, e.g. ID.791.2 where 2 is the item order number within the module.

The RandoNode is not expected to send back the OdmData object to the Portal for the methods that have been currently defined.

Attribute	dataType	Example data	Description
openClinicalData	string (longtext)	See the example in sections elsewhere below in this document. below. This contains XML with the base node as ODM.	This string contains the clinical data collected such as responses to the eligibility checklist questions.
openMetadata	string (longtext)	This is normally empty. This contains XML with the base node as ODM.	This string contains the form meta data. It is normally empty. The OPEN message API will read the metadata from the metadata files provided to the group and populate this object.

Transaction Overhead Classes

Overview

This section describes the classes used for the transaction/request overhead data. These request-related classes are passed as parameters to the request methods, and the response related classes are included in the response classes.

class OpenTxHeader

The OpenTxHeader is the common class that is included within the OpenRequest and returned in the OpenResponse classes. The OpenTxHeader contains the request tracking information. Groups are expected to return the OpenTxHeader back (within their response) whenever one of the methods are invoked. This involves simply setting the reference to the header object that is passed in the response object.

The following table describes the members of the OpenTxHeader class.

Attribute	dataType	Example data	Description
txGUID	v32	OPEN-080913-0000682	The unique request message id generated by the OPEN application. In case transaction from databases other than production the format of the txGUID will be OPEN-TST-080913-0000682. The database name follows the string OPEN. In the example, it indicates that the tx is coming from CTSUTST database.
timeStamp	date	2008-04-15T19:31:26.453Z	The date time that this request is generated by the OPEN or the calling application.
targetGroup	v20	NSABP ECOG SWOG	The cooperative group to which the OpenRequest is addressed to
txType	v32		Transaction type. Not used at this time. It is there for future use.
sourceComponent	v32	PORTAL OPENDB RSS	This is the application or CTSU resource that is trying to consume the RandoNode service. The example values are PORTAL, OPENDB, and RSS etc...
isTest	boolean	TRUE FALSE	The isTest is intended to send test registration during development phase to the group RandoNode. If isTest is TRUE, the request is for test registration. For processing the registration in production mode, groups should verify that the isTest is FALSE. This is a redundant ability, in addition to the registerTest method that the groups are implementing. Having isTest will be helpful in other testing situations using methods such doCredential, doValidate etc.
otherValues	string (longtext)	XML	This is included to provide a mechanism to pass additional parameters without having to modify the class definition or API. It will be named-value pairs in XML. APIs to access this data easily will be provided.

class OpenRequest

This is a general purpose request class. This class contains the transactional information (OpenTxHeader) as well as the requested Operation.

In general, specific request classes are not necessary because the API method signatures allow for the passing of multiple parameters. However, if a specific request class is required, it will always include the OpenTxHeader object.

The following table describes the members of the OpenRequest class.

Attribute	dataType	Example data	Description
header	OpenTxHeader	See above	See the OpenTxHeader.
operation	v(32)	REGISTER VALIDATE CREDENTIAL TEST CHECK_DUP_PT RETRYnnn ManualRegistration DataTransfer	This is the intended operation expected out of the request. The example for operation includes REGISTER, VALIDATE, CREDENTIAL, TEST etc. This is redundant in addition to the web service methods doRegister, doCredential, doValidate etc... In case of CHECK_DUP_PT, the input request objects are similar to doValidation. Groups are expected to verify, based on the demographics data, that the patient is existing or not in their database. RETRynn indicates the registration is being submitted as a retry. nnn is the number of attempts. ManualRegistration and DataTransfer requests are only for information, randomization is not performed. Both the types are for sanding Registered data to group Randonode. Some of the groups support manual registration when system is not available, after OPEN comes back Registered data along with patient id and treatment assignment are entered in to the database using OPEN screens and is sent to group for synchronization. This type of request is called ManualRegistration. DataTransfer is to send registered request data to group at any later point of time in case group requests.
targetURL	v(100)	http://www.nsabp.pitt.edu/TestPrograms/RandoNode/RandoNode.asmx	The receiving group's RandoNode URL.
otherValues	string (longtext)	XML	This is included to provide a mechanism to pass additional parameters without having to modify the class definition or API. It will be named-value pairs in XML. APIs to access this data easily will be provided.

class OpenResponse

This class should be included in any special purpose response classes. The RegistrationResponse class is described below and includes a reference to this object. This object includes a reference to the originating OpenTxHeader object that was passed in the request. [Ravi, this is where the OpenTxHeader could be specified to be directly included in the specific response class.]

The response attributes should be filled in per the specification of each request.

Note that for some methods, a specific response class is not

Attribute	dataType	Example data	Description
header	OpenTxHeader		See OpenTxHeader. Note: You have to set the header object in this class to the OpenRequest.header object passed into your web method. This has to be set for all the web methods provided by RandoNode.

Attribute	dataType	Example data	Description
responseCode	v(32)	PROCESSED EXCEPTION READY NOT-READY	The overall status of this request, not necessary related to registration. The valid values are PROCESSED and EXCEPTION. In case of isAvaiable method, the valid response statsues are READY and NOT-READY.
responseText	v500	Exception Occurred processing the message. Invalid date.	The brief text to describe the operation result. This is mandatory in case of EXCEPTION.
responseDetailText	string (longtext)	Any relevant error dump.	The detailed description of the operation status. This is optional.
responseData	string (longtext)	XML	This is included to provide a mechanism to pass additional result data or to use this general purpose response class without having to implement a specific response class.

Specific Response Classes

Overview

Specific response classes are defined primarily as wrappers so that multiple objects can be returned from a method call. This avoids the copying of many attributes from objects into one big class. It simply includes references to the transaction overhead classes (e.g., OpenResponse) and to domain objects that need to be returned.

class RegistrationResponse

This is the return class for many of the OPEN Registration operation requests. The methods that are implemented that use this class as a response includes:

- doCredential
- doValidate
- doRegister
- doRegisterTest

This object has two members. **OpenResponse** has the unique message header, and the operation result.

The **OpenRegistration** has attributes that the Group RandoNode completes during the registration processing.

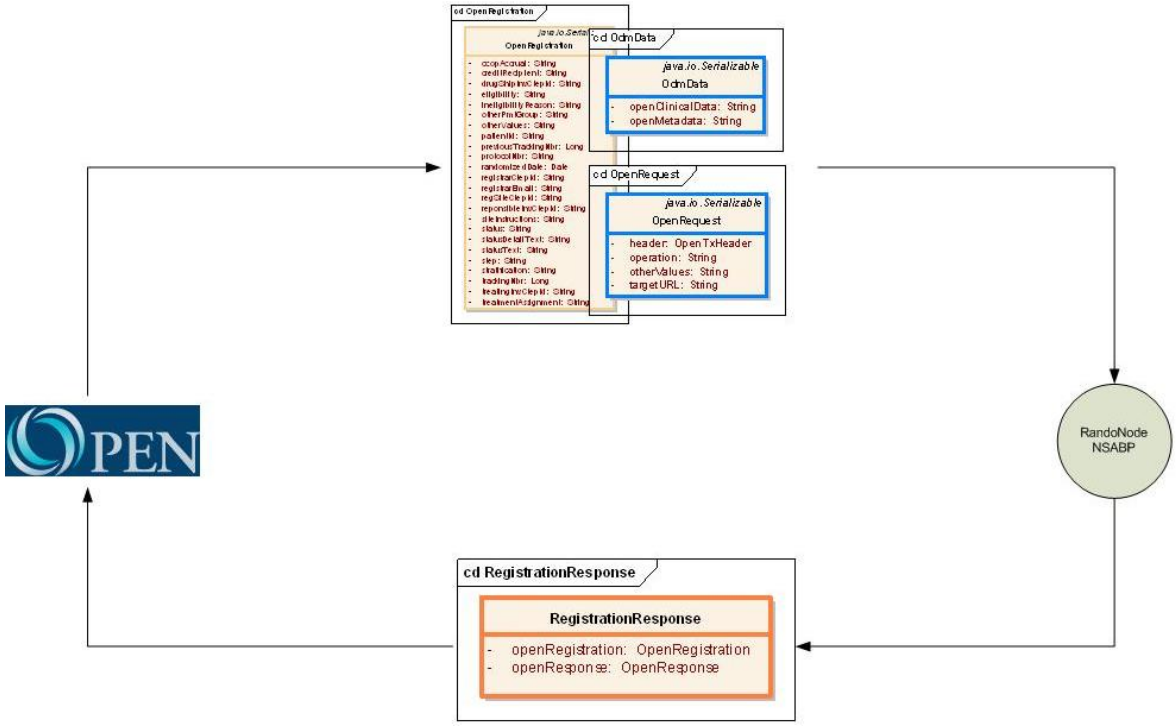
Attribute	dataType	Example data	Description
openResponse	OpenResponse	See above	See OpenResponse above
openRegistration	OpenRegistration	See above	See OpenRegistration above

doRegistration Request Flow Example

Flow Diagram

The object flow for a request from the OpenPortal to RandoNode when Open portal invokes the **doRegistration** method is shown in the following diagram.

Figure 2 Object flow when invoking doRegistration method



RandoNode jar

Overview

As part of the starter kit CTSU provide the library Randonode.jar/dll. The following are the packages included within the RandoNode jar/dll.

- com.westat.ctsu.open.node.framework – this is a framework suggested by OPEN on how to design the RandoNode such that the web service code is isolated from the registration business logic. This framework provides a clean separation between the web service input and the registration business logic. Please refer to the framework classes section above for more details.
- com.westat.ctsu.open.node – this package contains the RandoNode web service code, the web interface objects and the utility 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 – this package contains the classes that represent the ODM object model. The details of the ODM API are given below.
- com.westat.ctsu.open.node.domain – this package contains the persistent domain objects for working with hibernate. If the group decides to persist the input from OPEN for audit trail, or for

verification process, then the group can utilize this package with the persist package to save the data to the database (randonode schema defined by CTSU).

- com.westat.ctsu.open.node.persist – This is the hibernate persistent layer
- com.westat.ctsu.open.node.example – this package contains a few examples on how to test the local RandoNode, and how to retrieve the clinical data using the ClinicalDataUtil.java.

RandoNode Logic

When the request is invoked on the RandoNode side within the web service, you have the data in easy to use objects. It is at this point that the ODM API (described below) to access the data in the eligibility checklist can be utilized.

The logic in your RandoNode will be to create a registration record in your own patient registration system based on the OpenRegistration data and the eligibility checklist, and fill in some values in the OpenRegistration object to be returned. To return the OpenRegistration object, you will construct a RegistrationResponse object and place the OpenRegistration, and fill in some response values (e.g., PROCESSED, etc.) The response object is returned to the OPEN Portal for completion of the registration.

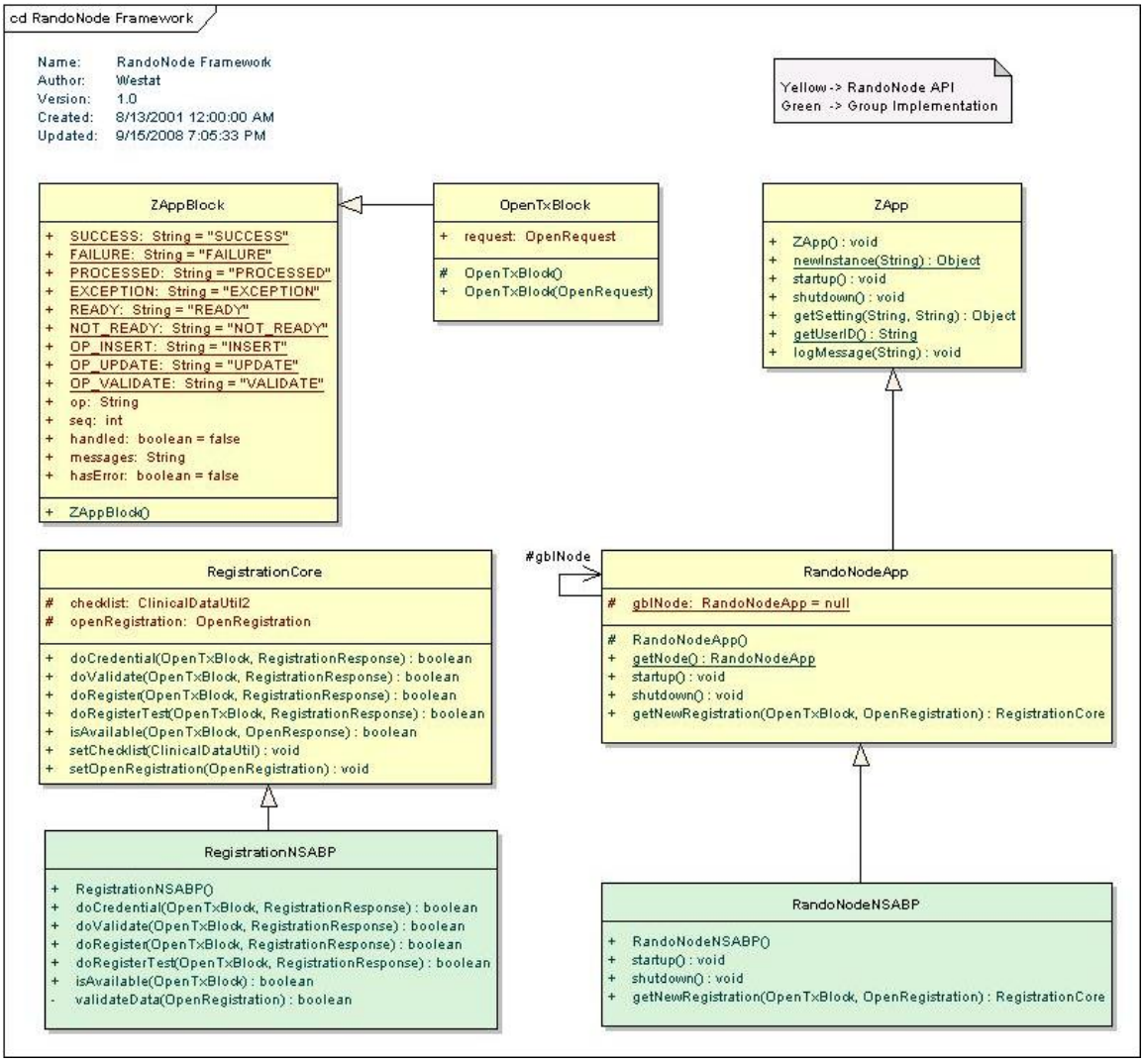
RandoNode Framework Classes

FrameWork Classes

Apart from the interface classes, the Randonode consists of few Framework classes.The framework classes reside in group Randonode and are not part of the data flow between OPEN and group Randonode. The classes facilitate the processing of randonode requests as well as ease of segregation of CTSU distributed classes for randonode from the group implemented classes. The following is the diagram of the Framework classes for ECOG. The framework classes implement methods that operate on the interface classes. The green colored classes will be maintained by OPEN and are part of the 'starterkit'. The yellow colored classes are to be implemented by the group.

More details to come.

RandoNode Framework Classes



OdmData Class

Overview

Many of the requests from Open Portal to randoNode include an odmData object. The Clinical data is transferred to RandoNode through the odmData object which consists of openClincialData and openMetaData as described in class OdmData. The openClincialData is a string that contains the clinical data in CDISC ODM format.

The architecture of the system separates the MetaData XML from the clinical data message XML. Both metadata and clinical data XML are CDISC ODM compliant XML strings with ODM as the root tag. The MetaData XML is installed as part of the RanodNode configuration.

This section describes metadata and Clinical Data XML format as well as the API that converts the clinical Data ODM string to objects to enable easier accessing of the data.

It is important to note that if you use the API, you do not need to understand all of the details of the XML.

MetaData XML

A separate MetaData XML file will be developed and delivered to the lead Group from the CTSU IT team for every version of the eligibility checklist form for each protocol. This XML file should become part of the Group RandoNode system installation and must be installed before the RandoNode can process registrations for that protocol version.

The ODM object will contain references to the metadata in that MetaData XML file. The following is an example of the MetaData XML. For brevity, the repeating tags have been removed from the examples.

Clinical Data XML

The Clinical Data will contain the actual responses to the eligibility checklist questions and will include references to the metadata in the specific version of a MetaData XML file. The following is an example of the Clinical Data XML. For brevity, the repeating tags have been removed from then examples.

Examples

MetaData XML

The following is an example of the MetaData XML in CDISC format.

```
<?xml version="1.0" encoding="UTF-8"?>
<ODM FileType="Snapshot" FileOID="meta.xml" CreationDateTime="2007-05-01T11:20:50"
     xmlns="http://www.cdisc.org/ns/odm/v1.3"  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://www.cdisc.org/ns/odm/v1.3 ODM-1-3-0.xsd">
  <Study OID="STUDY.NSABP-B-42">
    <GlobalVariables>
      <StudyName>NSABP-B-42</StudyName>
      <StudyDescription>A Clinical Trial to Determine the Efficacy</StudyDescription>
      <ProtocolName>NSABP-B-42</ProtocolName>
    </GlobalVariables>
    <MetaDataVersion OID="v.NSABP-B-42.1" Name="Version.1.0">
      <Protocol>
        <StudyEventRef StudyEventOID="SE.Registration.1" Mandatory="Yes"></StudyEventRef>
      </Protocol>
      <StudyEventDef OID="SE.Registration.1" Name="Registration" Repeating="No" Type="Scheduled">
        <FormRef FormOID="FORM.2494107.1.0" Mandatory="Yes"></FormRef>
      </StudyEventDef>
      <FormDef OID="FORM.2494107.1.0" Name="NSABP Protocol B-42 - Protocol Entry Form
         Worksheet" Repeating="No">
        <ItemGroupRef ItemGroupOID="IG.2" OrderNumber="0" Mandatory="Yes"></ItemGroupRef>
       ...
     </FormDef>
      <ItemGroupDef OID="IG.2" Name="Header" Repeating="No">
        <ItemRef ItemOID="ID.7" OrderNumber="0" Mandatory="Yes"></ItemRef>
        ...
      </ItemGroupDef>
      <ItemGroupDef OID="IG.3" Name="Protocol Eligibility" Repeating="No">
       ...
      </ItemGroupDef>
      <ItemDef OID="ID.7" Name="Patient Initials" DataType="text">
        <ExternalQuestion Dictionary="CaDSR" Version="4.0" Code="2001039"></ExternalQuestion>
      </ItemDef>
      <ItemDef OID="ID.8" Name="Site Name" DataType="text">
       .....
      </ItemDef>
      <CodeList OID="CL.32" Name="Medical Condition" DataType="text">
        <CodeListItem CodedValue="diabetes">
          <Decode>
            <TranslatedText xml:lang="en">DIABETES</TranslatedText>
          </Decode>
        </CodeListItem>
        <CodeListItem CodedValue="hypertension">
        ....
        </CodeListItem>
      </CodeList>
    </MetaDataVersion>
  </Study>
</ODM>
```

Clinical Data XML

The following is an example of the raw clinical data XML in CDISC format.

```
<?xml version="1.0" encoding="UTF-8"?>
<ODM FileType="Snapshot" FileOID="NSABP-B-42_2494107_1_0_clinical_151.xml"
     CreationDateTime="2007-05-14T02:21:07" xmlns="http://www.cdisc.org/ns/odm/v1.3"
     xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
     instance" xsi:schemaLocation="http://www.cdisc.org/ns/odm/v1.3 ODM-1-3-0.xsd">
  <AdminData>
    <User OID="USER.10124">
      <FirstName> Jasjeet Singh </FirstName>
      <LastName>Sangha </LastName>
      <Email>not available </Email>
      <Phone>(319) 272-2700 </Phone>
      <LocationRef LocationOID="LOC.FL035"></LocationRef>
    </User>
    <User OID="USER.500539">
    ...
    </User>
    <Location OID="LOC.FL035" Name="Cedars Medical Center">
      <MetaDataVersionRef StudyOID="STUDY.NSABP-B-42" MetaDataVersionOID="v.NSABP-B-42.1.0"
         EffectiveDate="2007-05-14"></MetaDataVersionRef>
    </Location>
  </AdminData>
  <ClinicalData StudyOID="STUDY.NSABP-B-42" MetaDataVersionOID="v.NSABP-B-42.1.0">
    <SubjectData SubjectKey="1">
      <AuditRecord>
        <UserRef UserOID="USER.500539"></UserRef>
        <LocationRef LocationOID="LOC.FL035"></LocationRef>
        <DateTimeStamp>2007-05-14T02:21:07 </DateTimeStamp>
        <SourceID>1 </SourceID>
      </AuditRecord>
      <InvestigatorRef UserOID="USER.10124"></InvestigatorRef>
      <SiteRef LocationOID="LOC.FL035"></SiteRef>
      <StudyEventData StudyEventOID="SE.Registration.1">
        <FormData FormOID="FORM.2494107.1.0">
          <ItemGroupData ItemGroupOID="IG.1" ItemGroupRepeatKey="1" TransactionType="Insert">
            <ItemData ItemOID="ID.1" Value="White"></ItemData>
            <ItemData ItemOID="ID.2" Value="Private Insurance"></ItemData>
            <ItemData ItemOID="ID.3" Value="NOT HISPANIC OR LATINO"></ItemData>
            <ItemData ItemOID="ID.4" Value="20856"></ItemData>
            <ItemData ItemOID="ID.5" Value="US"></ItemData>
          </ItemGroupData>
          <ItemGroupData ItemGroupOID="IG.2" ItemGroupRepeatKey="1" TransactionType="Insert">
            <ItemData ItemOID="ID.7" Value="DSU"></ItemData>
              ....
          </ItemGroupData>
        </FormData>
      </StudyEventData>
    </SubjectData>
  </ClinicalData>
</ODM>
```

Overview

Since the metadata and clinical data XML are CDSIC OMD v1.3 compliant, knowledge of ODM would generally be required for processing the ODM documents. This can become a bit complicated and requires understanding much more than is really necessary to effectively use the data.

In order to simplify the processing of the odmData object sent by OPEN, we have developed an object-oriented API. This API has been built on an object model derived from CDISC ODM. To help describe the API, two class diagrams have been included:

- The MetaData class diagram, and
- The Clinical Data class diagram.

The class diagrams are shown in Figure 1 and Figure 2 below. The classes do not contain attributes for all of tags in the CDISC ODM. They contain only classes corresponding to the ODM tags that OPEN system intends to use at this time. CTSU will be adding more classes if additional tags are added to the metadata and /or clinical data xml files.

Understanding the classes and the associated class methods will enable you to process the message and prepare a response without interacting with the XML directly.

Class Diagrams

A class diagram is used to give an overview of the system, showing the hierarchy of classes and their static relationships at varying levels of detail. Classes are represented as boxes in the diagrams. The enclosing box is divided into three sections:

- The topmost section provides the name of the class, and is often used as the identifier for the class;
- The middle section contains a list of attributes (structures) for the class; (the attribute in the class diagram maps into a column name in the data model and an attribute within the Java class);
- The bottom section lists the object's operations (methods).

The associations between classes are also indicated along with the following multiplicities:

Multiplicities	Description
0..1	Zero or one instance. The notation n..m indicates n to m instance
0..* or *	Zero to many; No limit on the number of instances (including asterisk (*) is used to represent a multiplicity of many.
1	Exactly one instance
1..*	At least one instance to many

Naming Conventions

The class name starts with an uppercase letter and an attribute starts with a lowercase letter. Names contain no underscores. If the name contains two words, then both words are capitalized, with no space between words. If an attribute contains two words, the second word is capitalized with no space between words.

The following notations (as shown in Fig.1 and Fig. 2) are used to indicate that a feature is public or private:

- "-" prefix signifies a private feature

- “+” signifies a public feature

In addition to the regular classes there are two utility classes:

- metaDataUtil, and
- clinicalDataUtil.

The utility classes have many useful methods to simplify the extraction of the data. A system generated javadoc and help files are also available for download. The documents describe the class definition, class attributes, and class methods. The XMLHelper referred in the class diagrams is a Document Object Model (DOM) object used to traverse the xml file.

The MetaDataUtil class extracts the registration form metadata in xml format, and puts the xml data content into the corresponding classes according to the xml elements. This class is designed to encapsulate the complexity of the ODM schema and to shield the end user from this complexity.

The ODM model assumes that a study's clinical data will consist of several kinds of entities. These include subjects, study events, forms, item groups, items, and annotations.

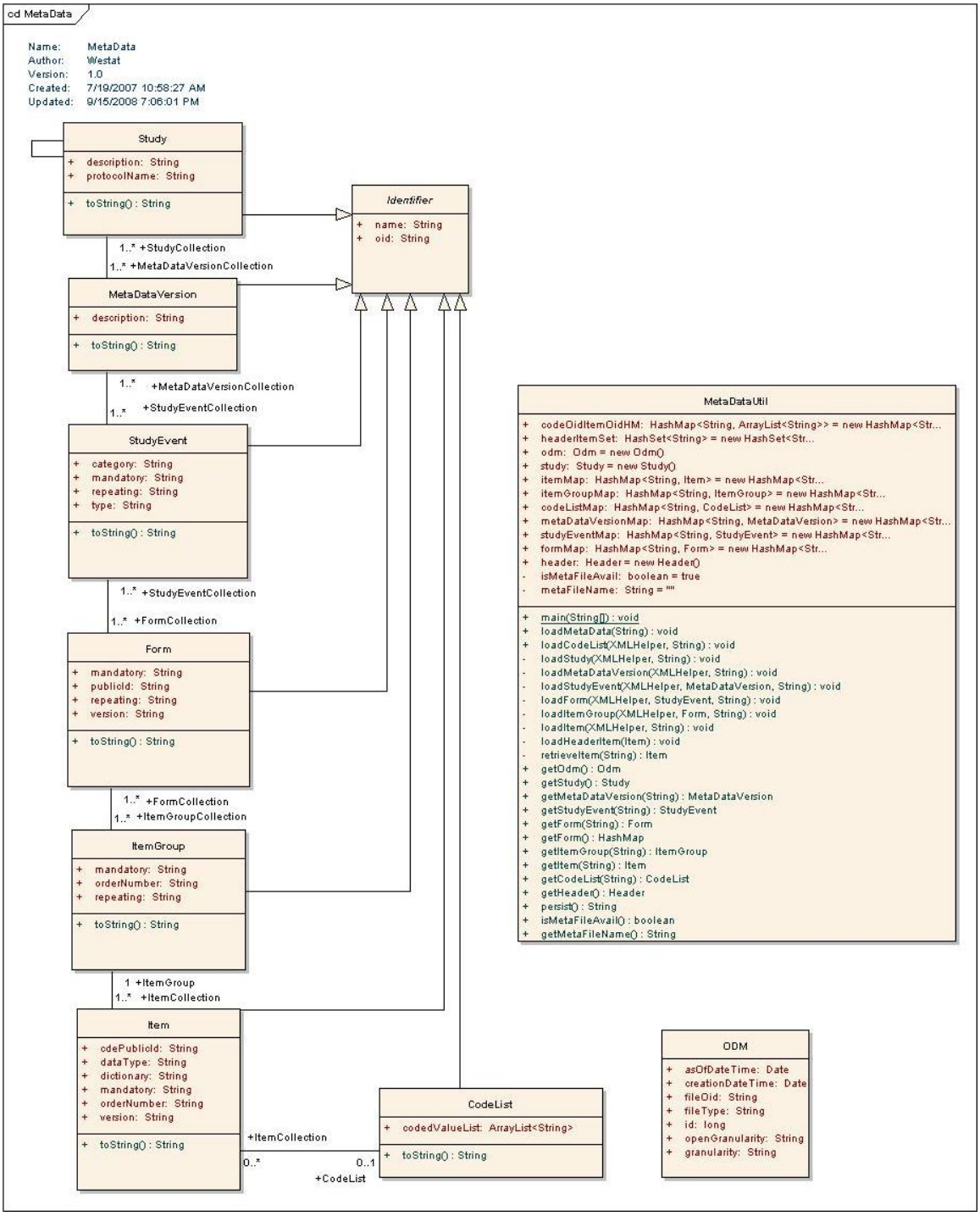
- An *item* is an individual clinical data item, such as a single systolic blood pressure reading. Items are collected together into item groups.
- An *item group* is a closely related set of items that are generally analyzed together. (Item groups are sometimes referred to as "records" and are associated with "panels" or "tables".) Item groups are aggregated into forms.
- A *form* is analogous to a page in a paper CRF book or electronic CRF screen. A form generally collects a set of logically and temporally related information. A series of forms is collected as part of a study event.
- A *study event* is a reusable package of forms usually corresponding to a study data-collection event.
- A study protocol design may include one or more planned subject visits at which data will be collected. Each planned visit may correspond to one or more ODM study events and ODM study events may be used for more than one subject visit.
- A *subject* is a patient participating in the study.

The metadata classes are designed to mimic the entities above. The ClinicalDataUtil class extracts the registration form responses in xml format and puts the xml data content into the corresponding classes according to the xml elements. This class is constructed in the similar fashion as the MetaDataUtil class and attempts to achieve the same encapsulation of the complexity. The ClinicalDataUtil class loads the MetaDataUtil object that corresponds to its form metadata.

The mapping between ODM tags and the classes is available in end of this document. This mapping allows one to identify source of the class member values within the XML string. In addition, the mapping tables provide the data type and length information for the variables.

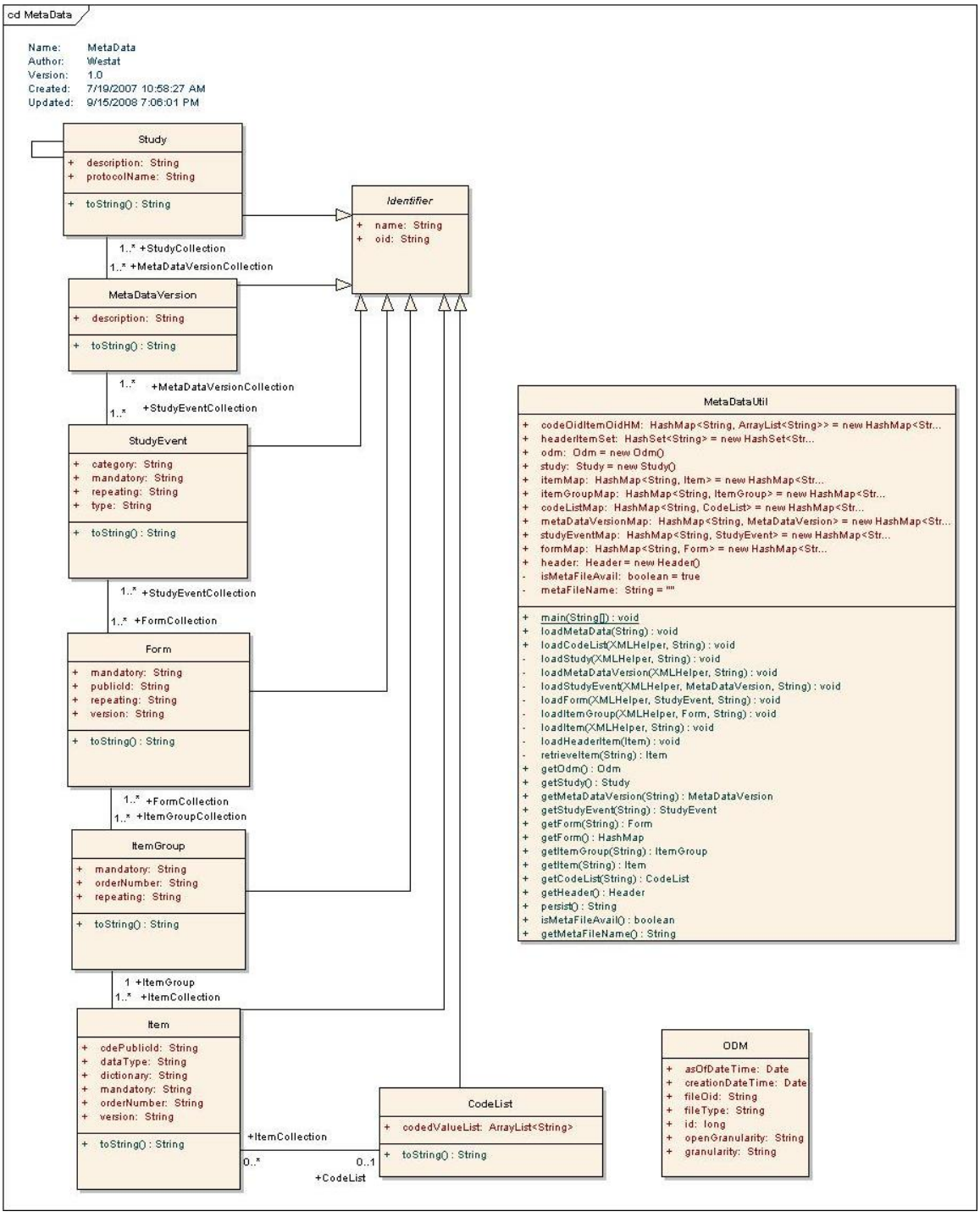
MetaData Class Diagram

Figure 3 Metadata Class Diagram



Clinical Data Class Diagram

Figure 4 Clinical Data Class Diagram



API Implementations

CTSU has developed the API in two languages.

- java version and
- C# (.NET) version.

The signatures of the classes used by both java and C# versions are the same as given the Figure1 and Figure 2. The CTSU is providing a **jar** file (java version) and **dll** (for C# version) along with the documentation.

The jar and dll have been developed in such a way that these can be dropped into the library folder of the RandoNode and can be invoked within the RandoNode application. The jar files and associated documents can be downloaded from www.ctsu.org/open (Note: the URL for the documents will be updated in future). At this site, there are two folders one for java and the other for .Net containing the downloadable files.

API Documentation

The document list will be updated soon. The documents for the different versions are given Table-2 below. All the documents required for java language is included in the OPEN_Request.zip. Similarly, all the documents required for java language is included in the RandoNodeExample.msi.

OPEN RandoNode Documentation

Version	File Name	Description
java, .NET	RandoNode_API.doc	This document on OPEN request processing.
java, .NET	ODM-1-3-0.xsd	CDISC ODM version 1.3 schema file.
java, .NET	ODM1-3-0-foundation.xsd	Schema containing definitions of the elements, attributes and data types that comprise ODM version 1.3
java, .NET	ODM1-3-0-Final.htm	Specification for the CDISC ODM Version 1.3. Available at CDISC_1.3 Specs
java	OPEN_Request.jar	Jar for the java version of the API
java	Readme.doc	Document containing the installation instruction.
.NET	Readme.pdf	Document containing the installation instruction.
java	javadoc folder within he OPEN_message.zip	Javadoc for the API
.NET	RandoNodeExample.msi	msi file when clicked install the .dll in the local machine.
.NET	Documentation.chm	Help file for .NET version of the API.
java, .NET	E1505_2555093_1_0_meta.xml	Sample meta data file for Registration worksheet for the ECOG protocol E1505
java, .NET	E1505_2555093_1_0_clinical_305.xml	Sample clinical data file containing data for Registration into ECOG protocol E1505.
java, .NET	NSABP-B-42_2494107_1_0_clinical_151.xml	Sample meta data file for Protocol Entry Form Worksheet for the NSABP protocol NSABP-B-42.
java, .NET	NSABP-B-42_2494107_1_0_meta.xml	Sample clinical data file containing data for Registration into NSABP protocol NSABP-B-42.
java, .NET	ODM_DB_schema_definition.txt	Database schema definition file for the creation of staging tables.
java, .NET	Example1.java	Example java file that shows how to extract the enrolling site and investigator CTEP ID from the clinical data
java, .NET	Example2.java	Example java file that shows how to extract the clinical data including the form name, question name and responses.

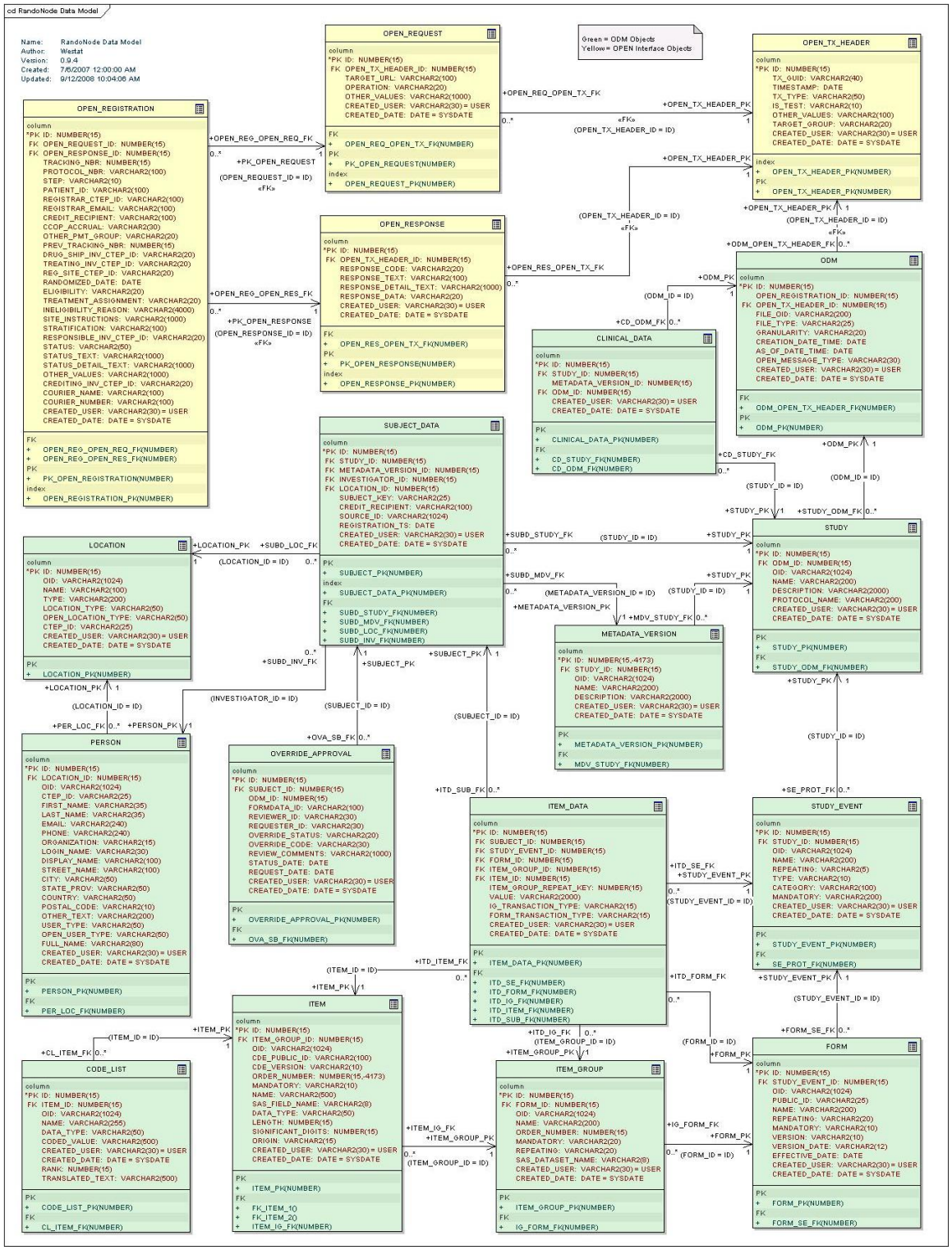
Version	File Name	Description
java, .NET	Example3.java	Example java file that shows how to construct the response to be sent to OPEN portal when a message is received.
java, .NET	Example4.java	Example java file that shows how to extract data from the metadata file (With no clinical data) using metaDataUtil.

Extraction to Staging Tables

The API also includes methods to extract the data from the metadata and the clinical data message and process them in real time. It is possible that groups may want to store the extracted data in staging tables.

The following diagram (Figure 5) shows the database schema to which the data from the metadata and the clinical data message can be persisted. The brown shaded boxes represent the metadata tables and the green shaded boxes represent the clinical data tables. All of the tables closely match the class diagrams. There is an extra table called ODM in the database schema that is not represented in the class diagrams. The class diagrams will be changed to include ODM class in the next release of the API. The table definition scripts that create the tables are also available from here. A hibernate utility is provided to persist the data and to extract the data based on a search criteria.

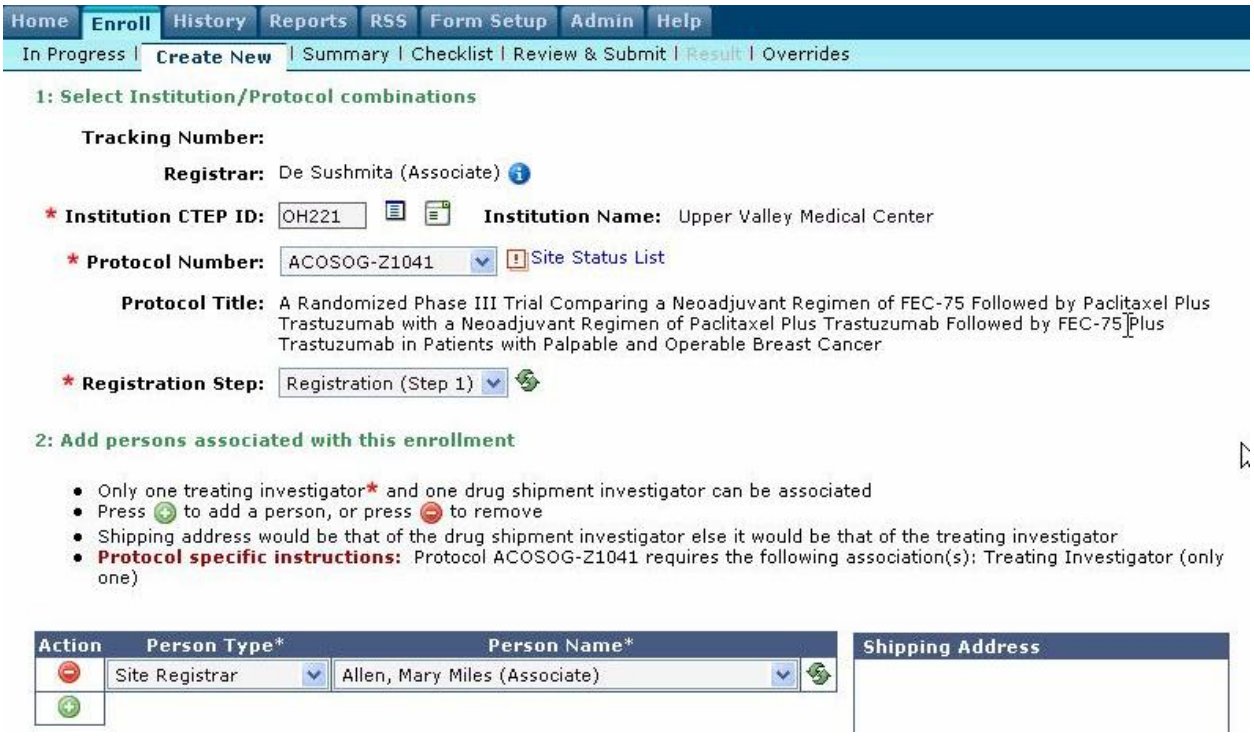
Figure 5


cd RandoNode Data Model
Name: RandoNode Data Model
Author: Westat
Version: 0.9.4
Created: 7/6/2007 12:00:00 AM
Updated: 9/12/2008 10:04:06 AM
Green = ODM Objects
Yellow = OPEN Interface Objects
OPEN_REQUEST
OPEN_TX_HEADER
OPEN_REGISTRATION
OPEN_RESPONSE
ODM
CLINICAL_DATA
SUBJECT_DATA
STUDY
LOCATION
METADATA_VERSION
PERSON
OVERRIDE_APPROVAL
ITEM_DATA
STUDY_EVENT
CODE_LIST
ITEM
ITEM_GROUP
FORM

Site Registrar Scenarios

During initial phase of OPEN production release, CTSU registrars will be responsible for entering registration data in OPEN on behalf of Site. On CreateNew screen, "Site Registrar" was added in the Person Type drop down (Figure 6) to make sure that OPEN captures the information of Site Person responsible for that registration request. During initial phase of production, all groups will have the option to select Site Registrar from the Person Type drop down.

Figure 6: Site Registrar



Group receives the Site Registrar detail from OPEN in clinical data XML. OpenRegistration.registrarCtepId, OpenRegistration.registrarEmail and User (within ODM, Admin Data section) contain the details of registrars. Different scenarios for Site Registrar have been explained below:

- **Site Registrar Enrolling Patient -** This situation will come up once Sites start enrolling patients. Then OpenRegistration.registrarCtepId and registrarEmail will contain Site Registrar's data.

```
- <openRegistration>
    <ccopAccrual>NO</ccopAccrual>
    <creditRecipient>CTSU</creditRecipient>
    <drugShipInvCtepId>NULL</drugShipInvCtepId>
    <eligibility>NULL</eligibility>
    <ineligibilityReason>NULL</ineligibilityReason>
    <otherPmtGroup>NULL</otherPmtGroup>
    <otherValues>NULL</otherValues>
    <patientId>NULL</patientId>
    <previousTrackingNbr>-99999999</previousTrackingNbr>
    <trackingNbr>29320</trackingNbr>
    <protocolNbr>NSABP-B-42</protocolNbr>
    <randomizedDate>2008-10-17 11:43:18.796</randomizedDate>
    <registrarCtepId>502271</registrarCtepId>
    <registrarEmail>geier@nsabp.pitt.edu</registrarEmail>
```

```
- <AdminData>
  - <User CtepId="502271" OID="USER.502271" OPENUserType="Associate" UserType="Other">
      <FullName>Geier, Michael</FullName>
      <FirstName>Michael</FirstName>
      <LastName>Geier</LastName>
      <Email>geier@nsabp.pitt.edu</Email>
      <Phone>412-383-1091</Phone>
      <LocationRef LocationOID="LOC.GA045" />
    </User>
```

- **CTSU Registrar Enrolling Patient and selects a Site Registrar** - This situation will arise during initial period of production. CTSU Registrar will enter the data on behalf of Site registrar and selects a Site Registrar. Then OpenRegistration.registrarCtepId & registrarEmail will contain Site Registrar's data.

```
- <openRegistration>
    <ccopAccrual>NO</ccopAccrual>
    <creditRecipient>CTSU</creditRecipient>
    <drugShipInvCtepId>NULL</drugShipInvCtepId>
    <eligibility>NULL</eligibility>
    <ineligibilityReason>NULL</ineligibilityReason>
    <otherPmtGroup>NULL</otherPmtGroup>
    <otherValues>NULL</otherValues>
    <patientId>NULL</patientId>
    <previousTrackingNbr>-99999999</previousTrackingNbr>
    <trackingNbr>29347</trackingNbr>
    <protocolNbr>NSABP-B-42</protocolNbr>
    <randomizedDate>2008-10-24 14:03:36.078</randomizedDate>
    <registrarCtepId>501881</registrarCtepId>
    <registrarEmail>sbennett@christianacare.org</registrarEmail>
```

```
- <AdminData>
  - <User CtepId="513646" OID="USER.513646" OPENUserType="Associate" UserType="Other">
      <FullName>De, Sushmita</FullName>
      <FirstName>Sushmita</FirstName>
      <LastName>De</LastName>
      <Email>Not available</Email>
      <Phone>Not available</Phone>
      <LocationRef LocationOID="LOC.CTSU" />
    </User>
  - <User CtepId="501881" OID="USER.501881" OPENUserType="Site Registrar" UserType="Other">
      <FullName>Bennett, Sheila E.</FullName>
      <FirstName>Sheila</FirstName>
      <LastName>Bennett</LastName>
      <Email>sbennett@christianacare.org</Email>
      <Phone>706-660-6404</Phone>
      <LocationRef LocationOID="LOC.GA045" />
    </User>
```

- **CTSU Registrar Enrolling Patient and does not select a Site Registrar** - This situation may arise if CTSU Registrar is entering data, but does not select a Site Registrar. Then OpenRegistration.registrarCtepId and registrarEmail will contain CTSU Registrar's data.

```
- <openRegistration>
    <ccopAccrual>NO</ccopAccrual>
    <creditRecipient>CTSU</creditRecipient>
    <drugShipInvCtepId>NULL</drugShipInvCtepId>
    <eligibility>NULL</eligibility>
    <ineligibilityReason>NULL</ineligibilityReason>
    <otherPmtGroup>NULL</otherPmtGroup>
    <otherValues>NULL</otherValues>
    <patientId>NULL</patientId>
    <previousTrackingNbr>-99999999</previousTrackingNbr>
    <trackingNbr>29338</trackingNbr>
    <protocolNbr>NSABP-B-42</protocolNbr>
    <randomizedDate>2008-10-24 15:02:49.734</randomizedDate>
    <registrarCtepId>500539</registrarCtepId>
    <registrarEmail>ravir@westat.com</registrarEmail>
```

```
- <AdminData>
  - <User CtepId="500539" OID="USER.500539" OPENUserType="Associate" UserType="Other">
      <FullName>Rajaram, Ravi</FullName>
      <FirstName>Ravi</FirstName>
      <LastName>Rajaram</LastName>
      <Email>ravi.rajaram@westat.com</Email>
      <Phone>301-294-2045</Phone>
      <LocationRef LocationOID="LOC.CTSU" />
    </User>
```

Mapping Classes to ODM Tags

The Mapping between the class members and the ODM xml elements and brief description is given in Table-1 below.

Table 1 Study Class

Class.member	ODM XML Element path	Datatype/ Length	Description
Study.oid	ODM/Study.OID	v25	Unique identifier for the study. The oid (short form for ODM ID?) contains the protocol number as defined in PIO (protocol information Office) database. The format of the study OID is Study.protocol_number.
Study.name	ODM/Study/GlobalVariables.StudyName	v200	This is the protocol number of the study.
Study.description	ODM/Study/GlobalVariables.StduyDescription	v2000	The title of the study
Study.protocolName	ODM/Study.	v200	The protocol number for the study. As per CDISC ODM, each study can have multiple protocols. For OPEN purposes, the protocol and study will mean the same. The protocol and study will be used interchangeably in this document.
Study.metaDataVersionList	ODM/MetaDataVersion	-	This indicates the version of the metadata in the OPEN database. This facilitates the handling of the protocol versions which require a changes in metadata.

Table 2 MetaDataVersion Class

Class.member	ODM XML Element path	Datatype /length	Description
MetaDataVersion.oid	ODM/Study/MetaDataVersion.OID	v100	The OID contains the study name, and the study version number. The format is v.<protocol_number>_<form_CADSR_public_ID>_<OPEN_Version_date_for the form>_step<step number>_meta.xml. This is equal to the attribute FileOID of the ODM tag. Though by default the OPEN_version_date defaults to a date string (yymmdd), it can be set to any string as desired the cooperative group.
MetaDataVersion.name	ODM/Study/MetaDataVersion.Name	v25	The version number of the meta data. The format is 'v'.protocol_number.version_number
MetaDataVersion.description	ODM/Study/MetaDataVersion.Description	v2000	Brief description of this version.
MetaDataVersion.studyEvent List	ODM/Study/MetaDataVersion/StudyEventRef	-	Protocol can have many study events. A StudyEvent describes a type of study event that takes place during a study. A scheduled event is one that is expected to occur for each subject as part of the ordinary progress of the study. For OPEN we will be dealing with REGISTRATION events corresponding to different steps.
MetaDataVersion.study	ODM/Study	-	Reverse association to Study class.

Table 3 StudyEvent Class

Class.member	ODM XML Element path	Datatype/ Length	Description
StudyEvent.oid	ODM/Study/MetaDataVersion/StudyEventDef.OID	v25	Unique identifier for a study event. The format is 'SE'.studyeventname.
StudyEvent.name	ODM/Study/MetaDataVersion/StudyEventDef.Name	v25	Study event name.

Class.member	ODM XML Element path	Datatype/ Length	Description
StudyEvent.repeating	ODM/Study/MetaDataVersion/StudyEventDef.Repeating	v3	Whether the study event can be repeated within the study. The Repeating flag indicates that this type of study event can occur repeatedly within the containing study
StudyEvent.type	ODM/Study/MetaDataVersion/StudyEventDef.Type	v10	This defines the study event type. The value can be Scheduled, Unscheduled or Common.
StudyEvent.category	ODM/Study/MetaDataVersion/StudyEventDef.Category	v100	The study event category. The Category attribute is typically used to indicate the study phase appropriate to this type of study event. For example, Step 1 registration.
StudyEvent.mandatory	ODM/Study/MetaDataVersion/StudyEventDef.mandatory	v3	The value can be either Yes or No. The value defines if this study event is mandatory. For Registration events for OPEN, it is normally equal to 'Yes'.
StudyEvent.formList	ODM/Study/MetaDataVersion/StudyEventDef.FormRef	-	Form describes a type of form that can occur in a study.
StuydEvent.metaDataVersion	ODM/Study/MetaDataVersion	-	Reverse association to MetaDataVersion class.

Table 4 Form Class

Form.oid	ODM/Study/MetaDataVersion/FormDef.OID	v25	This contains the public id as defined by CaDSR. The format is 'FORM'.cadsr_public_id
Form.name	ODM/Study/MetaDataVersion/FormDef.Name	v200	This is the name of the form as defined in CaDSR.
Form.publicId	SubString of the ODM/Study/MetaDataVersion/FormDef.OID	v100	This is derived from the form.oid.
Form.version	SubString of the ODM/Study/MetaDataVersion/FormDef.OID	v10	For OPEN, this corresponds to the meta data version.
Form.repeating	ODM/Study/MetaDataVersion/FormDef.Repeating	v3	The Repeating flag indicates that this type of form can occur repeatedly within the containing study event. This defines if the form can be used more than once under the same study and study event. For example, the form can be used for 1st visit, 2nd visit, and so on.

Form.mandatory	ODM/Study/MetaDataVersion/FormRef.Mandatory	v3	Flag to indicate where the form is mandatory or not. For OPEN, this always 'Yes'.
Form.itemGroupList	ODM/Study/MetaDataVersion/FormDef/ItemGroupRef	-	This contains all the item groups under this form. An item group is logically grouped questions under a section. The Item group corresponds to the module defined in caDSR form builder.
Form.studyEvent	ODM.Study/MetaDataVersion/StudyEventDef	-	Reverse association to StudyEvent class.

Table 5 ItemGroup Class

Class.member	ODM XML Element path	Datatype/ Length	Description
ItemGroup.oid	ODM/Study/MetaDataVersion/ItemGroupRef.OID	v25	This identifies an item group within the form. The format is 'IG'.<module_name>
ItemGroup.orderNumber	ODM/Study/MetaDataVersion/ItemGroupRef.OrderNumber	number	This defines where this item group occurs within the form. In other words the display sequence number of the item group within a form.
ItemGroup.name	ODM/Study/MetaDataVersion/ItemGroupDef.Name	v500	The name of the item group. Equal to the Module name as defined in caDSR form builder.
ItemGroup.mandatory	ODM/Study/MetaDataVersion/ItemGroupRef.Mandatory	v3	Flag to indicate where the item group is mandatory or not.
ItemGroup.repeating	ODM/Study/MetaDataVersion/ItemGroupDef.Repeating	v3	This defines if the item group contains questions (records) that can be repeated within the form. An example of repeating item group is a tabular module containing lab data such as lab name, value, unit of measurement etc. Within this repeating item group, there may be several records corresponding to several lab tests.
ItemGroup.itemList	ODM/Study/MetaDataVersion/ItemGroupDef.ItemRef	-	This defines the item within an item group. For example, the question patient race is under the Demographic.
ItemGroup.form	ODM.Study/MetaData/FormDef	-	Reverse association to Form class.

Table 6 Item Class

Class.member	ODM XML Element path	Datatype/ Length	Description
Item.oid	ODM/Study/MetaDataVersion/ItemDef.OID	v25	OID of Item or Question within a form. The format is 'ID'. <caDSR public id (CDE ID) for the question>.
Item.name	ODM/Study/MetaDataVersion/ItemDef.Name	v500	An Item describes a type of item that can occur within a study. Item properties include name, data type, data size, measurement units, range or code list restrictions, and several other properties
Item.orderNumber	ODM/Study/MetaDataVersion/ItemRef.OrderNumber	number	This defines the item display order (or occurrence order) within the item group.
Item.mandatory	ODM/Study/MetaDataVersion/ItemRef.Mandatory.	v3	Flag to indicate where the item is mandatory or not. The mandatory field value can be either Yes or No. It defines that this question has to have a response.
Item.dataType	ODM/Study/MetaDataVersion/ItemDef.DataType	v50	This field defines the question response data type. It can be integer, string, date, etc. See ODM document for more details of data type at ODM
Item.version	ODM/Study/MetaDataVersion/ItemDef/ExternalQuestion.Version	v10	The question version as defined within CaDSR.
Item.code	ODM/Study/MetaDataVersion/ItemDef/ExternalQuestion.Code	v100	The question unique id given by CaDSR. It is the CDE ID (Common Data Element ID) of the question as defined in caDSR.
Item.dictionary	ODM/Study/MetaDataVersion/ItemDef/ExternalQuestion.Dictionary	v10	This value defines who the question originator is. For example, if the question is originating from CaDSR, the values will be equal to caDSR. If the question is defined with OPEN to take care of a custom requirement, then it will be equal to OPEN.
Item.codeListList	ODM/Study/MetaDataVersion/ItemDef/CodeListRef	-	This defines a discrete set of permitted values for an item. For example, the race question can be one or more of the following values: White, Black, Other, etc.

Class.member	ODM XML Element path	Datatype/ Length	Description
Item.itemGroup	ODM.Study/MetaDataVersion/itemGroupDef	-	Reverse association to ItemGroup class.

Table 7 Location Class

Class.member	ODM XML Element path	Datatype/ Length	Description
Location.oid	ODM/AdminData/Location.OID	v25	A physical location -- typically a clinical research site or a sponsor's office. The format is 'LOC.'CTEP_ID of the institution.
Location.name	ODM/AdminData/Location.Name	v100	The name of the location. For example, the name value can be "Cedars Medical Center".
Location.locationType	ODM/AdminData/Location.LocationType	v10	This identifies the location type. The allowed values are Sponsor, Site, CRO, Lab or Other.
Location.CtepId	ODM/AdminData/Location.CtepId	v10	This is the CTEP iD of the institution.
Location.OPENLocationType	ODM/AdminData/Location. OPENLocationType	v50	This attribute serves the similar purpose as the OPENUserType. This attribute describes the type of facility, for example, the location is a doctor office, or a hospital.

Table 8 User Class

Class.member	ODM XML Element path	Datatype/ Length	Description
User.oid	ODM/AdminData/User.OID	v25	User contains Information about a specific user of a clinical data collection system. This may be an investigator, a CRA who is enrolling the patient. Study subjects are *not* users in this sense. This contains the CTSU unique id. The format of OID is 'USER'.CTEP_INVESTIGATRO_ID or UIN_NUMBER
User.locationOid	ODM/AdminData/User/LocationRef.LocationOID	v25	This is where the user is located.
User.firstName	ODM/AdminData/User/FirstName	v50	User's first name.

Class.member	ODM XML Element path	Datatype/ Length	Description
User.lastName	ODM/AdminData/User/LastName	v50	User's last name.
User.email	ODM/AdminData/User/Email	v50	User's email address.
User.phone	ODM/AdminData/User/Phone	v50	User's phone number.
User.userType	ODM/AdminData/User.UserType	v15	This defines the type of the user. This is optional. The allowable values are sponsor, investigator, lab or other. This is defined by ODM.
User.streetName	ODM/AdminData/User/Address/StreetName	v100	The street address part of a user's postal address
User.city	ODM/AdminData/User/Address/City	v50	The city name part of a user's postal address.
User.stateProv	ODM/AdminData/User/Address/StateProv	v50	The state or province name part of a user's postal address.
User.country	ODM/AdminData/User/Address/Country	v2	The country name part of a user's postal address. This must be represented by an ISO 3166 two-letter country code. Example: FR for France, JP for Japan.
User.postalCode	ODM/AdminData/User/Address/PostalCode	v10	The postal code part of a user's postal address.
User.otherText	ODM/AdminData/User/Address/OtherText	v200	Any other text needed as part of a user's postal address.
User.CtepId	ODM/AdminData/User.CtepID	V5	CTEP ID of the person
User.OPENUserType	ODM/AdminData/User.OPENUserType	V50	User type of the person as defined in RSS for the protocol. Example values are Treating Investigator, Drug Shipment Investigator, Associate Etc..

Table 9 ClinicalData Class

Class.member	ODM XML Element path	Datatype/ Length	Description
ClinicalData.studyOid	ODM/ClinicalData.StudyOID	v25	Same as the ODM/Study.OID.
ClinicalData.metaDataVersionOid	ODM/ClinicalData.MetaDataVersionOID	v100	Same as the ODM/Study/MetaDataVersion.OID.

Class.member	ODM XML Element path	Datatype/ Length	Description
ClinicalData.subjectDataList	ODM/ClinicalData/SubjectData	-	The data for a subject. Each clinical file represents a registration form for an accrual. Note: Subject attributes (such as date of birth, sex, responsible investigator, and so on) should be treated as if they were clinical data. Thus they can be changed as needed, and are subject to normal auditing processes.

Table 10 SubjectData

Class.member	ODM XML Element path	Datatype/ Length	Description
SubjectData.subjectKey	ODM/ClinicalData/SubjectData.SubjectKey	v25	This subject key is the unique key generated by CTSU to identify the registration. This id does not generated based on the Study, or StudyEvent. In the future, we can use Study name, study event and the patient natural data to identify the subject for this clinical data.
SubjectData.auditRecod.sourceId	ODM/ClinicalData/SubjectData.AudtiRecord.SourceId	v25	This will be equal patient ID if available, For multi step registrations this can be used to link the patient across registrations.
SubjectData.auditRecod.person	ODM/ClinicalData/SubjectData.AudtiRecord.UserRef.UserOID	-	This is a link to the user date under the AdminData section. This is the person who fills out the registration form.
SubjectData.auditRecord.location	ODM/ClinicalData/SubjectData.AudtiRecord.SiteRef.LocationOID	-	This is a link to the location data under the AdminData section. This is the site where the application is filled.
SubjectData.auditRecord.dateTimeStamp	ODM/ClinicalData/SubjectData.AudtiRecord.DataTimeStamp	v50	This is the datetime when the registration is submitted to the group for approval.
SubjectData.investigator	ODM/ClinicalData/SubjectData. InvestigatorRef.UserOID	-	This is the link to a user defined in the AdminData section. This identifies the treating physician.
SubjectData.site	ODM/ClinicalData/SubjectData/SiteRef.LocationOID	-	This a link to the location data under the AdminData section. This is the site where the subject is.

Class.member	ODM XML Element path	Datatype/ Length	Description
SubjectData.studyEventDataList	ODM/ClinicalData/SubjectData.StudyEventData	-	This links the patient information to a unique study event.

Table 11 StudyEventData Class

Class.member	ODM XML Element path	Datatype/ Length	Description
StudyEventData.studyEventID	ODM/ClinicalData/SubjectData.StudyEventData.StudyEventOID	v25	This is the unique id for the studyEventData
StudyEventData.subjectData	ODM/ClinicalData/SubjectData	-	Reverse association to subjectData.
StudyEventData.formDataList	ODM/ClinicalData/SubjectData/StudyEventData/FormData	-	This contains all the form data within the studyEventData.
StudyEventData.metadata	ODM/Study/MetaDataVersion/StudyEventDef	-	This is the link to studyEvent metadata.

Table 12 FormData Class

Class.member	ODM XML Element path	Datatype/ Length	Description
FormData.oid	ODM/ClinicalData/StudyEventData.FormData.FormOID	v25	This is the same form id as defined in the meta data.
FormData.itemGroupDataList	ODM/ClinicalData/StudyEventData.FormData.ItemGroupData.ItemGroupOID	-	This contains all the item group data within the form.
FormData.studyEventData	ODM/ClinicalData/SubjectData.StudyEventData.StudyEventOID	-	Reverse association to studyEventData
FormData.metadata	ODM/Study/MetaDataVersion/FormDef	-	This is the link to form metadata.

Table 13 ItemGroupData Class

Class.member	ODM XML Element path	Datatype/ Length	Description
ItemGroupData.oid	ODM/ClinicalData/StudyEventData/FormData/ItemGroupData.ItemGroupOID	v25	This links to the same item group in the meta data.

Class.member	ODM XML Element path	Datatype/ Length	Description
ItemGroupData.itemGroupRepeatKey	ODM/ClinicalData/StudyEventData/FormData/ItemGroupData.ItemGroupRepeatKey	number	This identifies the repeating position of the item group within the same form. For example, tabular data that can be represented in a tabular form are designed as repeating question groups. Every row in the table is assigned a unique Repeat key starting from 1.
ItemGroupData.transactionType	ODM/ClinicalData/StudyEventData/FormData/ItemGroupData.TransactionType	v20	To identify if the data transmitted within this form is new, update or should be deleted. When the form is submitted for the first time, it should be insert.
ItemGroupData.itemDataList	ODM/ClinicalData/StudyEventData/FormData/ItemGroupData.ItemData	-	This contains all the item data under one item group data.
ItemGroupData.formData	ODM/ClinicalData/StudyEventData/FormData	-	Reverse association to formData.
ItemGroupData.metadata	ODM/Study/MetaDataVersion/ItemGroupDef	-	This is the link to itemGroup metadata.

Table 14 ItemData Class

Class.member	ODM XML Element path	Datatype/ Length	Description
ItemData.oid	ODM/ClinicalData/StudyEventData/FormData/ItemGroupData/ItemData.ItemOID	v25	This is the unique ID for the item.
ItemData.value	ODM/ClinicalData/StudyEventData/FormData/ItemGroupData/ItemData.Value	v200	The response for the question. It is 'The Data'.
ItemData.metadata	ODM/Study/MetaDataVersion/ItemDef	-	This links to the meta data in order to get the meta data information of this item.
ItemData.itemGroupData	ODM/ClinicalData/StudyEventData/FormData/ItemGroupData	-	Reverse association to itemGroupData

Table 15 OverrideApproval Class

Class.member	ODM XML Element path	Datatype/ Length	Description
OverrideApproval.id	N/A	Number	This is the unique id generated by the database when the row is inserted.

Class.member	ODM XML Element path	Datatype/ Length	Description
OverrideApproval.reviewerId	ODM/ClinicalData/StudyEventData/FormData/OverrideApproval .reviewerId	V30	This is the CTEP/NCI id of the override reviewer.
OverrideApproval.requesterId	ODM/ClinicalData/StudyEventData/FormData/OverrideApproval .requesterId	V30	This is the CTEP/NCI id of the override reviewer.
OverrideApproval.overrideStatus	ODM/ClinicalData/StudyEventData/FormData/OverrideApproval .overrideStatus	V20	This is the override request status. It should always be Approved.
OverrideApproval.reviewComments	ODM/ClinicalData/StudyEventData/FormData/OverrideApproval .reviewComments	V1000	The reviewer comments if available.
OverrideApproval.statusDate	ODM/ClinicalData/StudyEventData/FormData/OverrideApproval .statusDate	Date	The override request review date.
OverrideApproval.requestDate	ODM/ClinicalData/StudyEventData/FormData/OverrideApproval .requestDate	Date	The override request date.

Examples

Examples-java

Few examples have been created to make it easier to understand the working of the API.

OPENPortalProxy

The following example shows the OPENPortalProxy which acts a a proxy for OPEN open portal. The OPENPortalProxy reads the OpenRequestObjects.xml and a corresponding metadata xml file NSABP-B-42_2494107_080320_step1_meta.xml. The OpenRequestObjects.xml should be available in the root folder of the project and the NSABP-B-42_2494107_080320_step1_meta.xml should be available under <CTSU_HOME>/RandoNode/meta folder. Click on the following objects to open the NSABP-B-42_2494107_080320_step1_meta.xml and OpenRequestObjects.xml


OpenRequestObjects.xml


NSABP-B-42_2494107_080320_step1_meta.xml

```
package com.westat.ctsu.open.node.example;

import java.io.FileReader;
import java.util.Scanner;
import com.westat.ctsu.open.node.OdmData;
import com.westat.ctsu.open.node.OpenRegistration;
import com.westat.ctsu.open.node.OpenRequest;
import com.westat.ctsu.open.node.OpenResponse;

import com.westat.ctsu.open.node.RandoNode;
import com.westat.ctsu.open.node.RegistrationResponse;
import com.westat.ctsu.open.node.RegistrationXML;



/**
 * Sample program to invoke the RandoNode method using the CTSU provided xml file
 * OpenRequestObjects. This can be used a testing
 * tool to test the RandoNode locally
 */
public class OpenPortalProxy {

    public static void main(String[] args) {
       /**
         * Example to show how to use the xml file to invoke
         * RandoNode web service. The xml ile contains data for the three objects
         * OpenRequest, OpenRegistration and OdmData required to invoke the doRegister method
         * The xml file has the same structure as the objects.
         */
        if(args.length < 1){
            System.out.println("usage: Example1 OpenRequestObjectsXML");
            System.exit(0);
        }
        try {
```

```
            FileReader reader = new FileReader(args[0]);
            Scanner in = new Scanner(reader);
            StringBuffer openRequestObjectsBuf = new StringBuffer();
            while (in.hasNextLine()){
                openRequestObjectsBuf.append(in.nextLine());
            }

            in.close();

            RegistrationXML toObj = new RegistrationXML();
            toObj.convertOpenRequestObjects(openRequestObjectsBuf.toString());
            RandoNode randonode = new RandoNode();

            OpenRequest openRequest = toObj.getOpenRequest();
            OpenRegistration openRegistration = toObj.getOpenRegistration();
            OdmData odmData = toObj.getOdmData();

            /**
             * Using the objects created from the XML string, call the web service that
             * accepts the objects as input
             */
            RegistrationResponse registResponse = randonode.doRegister(openRequest,
                    openRegistration, odmData);
           OpenRegistration openreg =registResponse.getOpenRegistration();
           OpenResponse openres =registResponse.getOpenResponse();
           System.out.println("OpenPortalProxy - reponse status= " + openres.getResponseCode());
           System.out.println("OpenPortalProxy - Reg status= " + openreg.getStatus());
           System.out.println("OpenPortalProxy - patient id= " + openreg.getPatientId());
           System.out.println("OpenPortalProxy - tretament arm= " +
     openreg.getTreatmentAssignment());


        }
        catch (Exception e) {
              e.printStackTrace();
        }
    }
}
```

RandoNode Configuration

This example shows how to configure the RandoNode to initialize the registration class for specific protocol.

```
package com.NSABP.ctsu.open.node;

import com.westat.ctsu.open.node.OpenRegistration;
import com.westat.ctsu.open.node.framework.OpenTxBlock;
import com.westat.ctsu.open.node.framework.RandoNodeApp;
import com.westat.ctsu.open.node.framework.RegistrationCore;
import com.westat.kjdk.app.Logger;

/**
 * This is the core class that defines the API for the overall application
 */
public class RandoNodeNSABP extends RandoNodeApp {
  /**
```

```
   * Constructor saves reference to the singleton
   */
  public RandoNodeNSABP() {
  }

  public void startup() {
    /**
     * do any start up work for the RandoNode, example open DB connection pool
     */
    Logger.log("RandoNode", "RandoNodeApp startup");
  }

  // Hook for the shutdown of the services
  public void shutdown() {
    Logger.log("RandoNode", "RandoNodeApp shutdown");
  }

  /**
   * Each node has to instantiate the specific sub-class of Registration
   * that will be used.
   */
  public RegistrationCore getNewRegistration(OpenTxBlock tx,
                                             OpenRegistration openRegistration) {
    /**
     * the group can and may have different Registration Handler for
     * the protocols. Below is an example on how to return the Registration
     * Handler for NSABP-B-42.
     * NOTE: check for null before proceed
     */
    /**
     * NOTE: openRegistratoion object can be null if the web method does
     *       not require OpenRegistration object as input object.
     *       The isAvailable method accepts only OpenRequest object,
     *       the openRegistration input object is null when this method
     *       is invoked.
     */
      if((openRegistration != null) &&
         (openRegistration.getProtocolNbr().equalsIgnoreCase("NSABP-B-42"))){
        //RegistrationNSABPB42 reg = new RegistrationNSABPB42();
       return (new RegistrationNSABPB42());
      } else {
         return (new RegistrationNSABP());
      }
    //return reg;
  }

  // Define other virtual functions here
  //
}
```

Data Extraction

This example shows the data extraction wthin the doRegister method of RegistrationNSABPB42.java.The example has several sections showing different ways of extracting data for simple questions, repeating module questions, and perisisting data to the RandoNode schema. An annotatated B 42 form is available

at te end of the document shows the question public id(in red), module name (in blue) as well as data. extract the data for the following questions:

- Ethnicity
- Date of Most Recent Mammogram
- Did this patient receive any tamoxifen?

In addition, it sets the OpenResponse.status, OpenResponse.statusText, OpenRegistration.patientId, OpenRegistration.eligibility, OpenRegistration.status, OpenRegistration.statusText, OpenRegistration.statusDetailText.

Also for repeating module, example code extracts the following.

- Comments corresponding to Hypertension under Cardiac History module
- All the data from concomitant medications module corresponding to the agent category Lipid Lowering Drug

```
public boolean doRegister(OpenTxBlock tx,
      RegistrationResponse registrationResponse) {
    // check to make sure this is not testing, if it is testing, you
      //  should handle the data appropriately
    boolean successCode = false;
    OpenTxHeader header = registrationResponse.getOpenResponse().getHeader();
    OpenResponse response = registrationResponse.getOpenResponse();
    OpenRegistration registration = registrationResponse.getOpenRegistration();
    if (!header.getIsTest()) {
      // it is recommended that the group performs the registration data
      // validation first before it determines if the patient is eligible to
      // enroll in the protocol

      // first, check to make sure the meta data is available for this request
      // if it is available, proceed, if not return error code, missing
      // meta data file message
      if (checklist.metaDataUtil.isMetaFileAvail()) {
        // call to validate the data first
        if(doValidate(tx, registrationResponse)){

            /**
             * The clinical xml file has an attribute under ClinicalData
             * element that specify the metadata file associated
             * with this clinical responses. The MetaDataVersionOID attribute
             * contains the metadata file name
             */
           ClinicalData clinicalData = checklist.clinicalData ;
           // ClinicalDataUtil provides options to retrieve the question value

            String studyOID = clinicalData.studyOid;
             System.out.println("studyOID " + studyOID);

            // The example below shows how to get the question value based on question long name.
            // this returns all the question matches with the given long name.
           ArrayList<ItemData> ethnicity = checklist.getItemDataByQName("Ethnicity");
           // output the value for this example
           if (ethnicity.size() > 0) {
             System.out.print("Ethnicity - ");
             checklist.printItemDataValue(ethnicity);
           }
```

```
 // show how to get the question value using question oid (combination
 // of the String 'ID.' and the question public id
 // get the question Date of most recent Mammogram using public id
 ArrayList<ItemData> mammogram = checklist.getItemData("ID.2495106");
 if (mammogram.size() > 0) {
   System.out.print("Date of Most Recent Mammogram - ");
   checklist.printItemDataValue(mammogram);
 }

 // show how to get the question value based on module name and question id
 // get the question Did this patient receive any tamoxifen?
 ArrayList<ItemData> tamoxifen = checklist.getItemDataByMNameQId(
     "Adjuvant Hormonal Therapy History and Status", "ID.2203");
 if (tamoxifen.size() > 0) {
   System.out.print("Did this patient received any tamoxifen - ");
   checklist.printItemDataValue(tamoxifen);
 }

 // The following method extracts data for one row for the
 // repeating module concomitant medications
  getConcomitantMedications();
/**
  * Print out all the question/response
  */

 HashMap<String, ArrayList<ItemData>> itemMap = checklist.itemDataMap;
 Set<String> dataKeySet = itemMap.keySet();
 Iterator<String> dataIter = dataKeySet.iterator();
 while(dataIter.hasNext()){
   String itemKey = dataIter.next();
   ArrayList<ItemData> cdItemData = itemMap.get(itemKey);
   for(int index = 0; index < cdItemData.size(); index++){
     ItemData itemData = cdItemData.get(index);
     //System.out.println(itemData.toString());
   }
 }

// please replace the below dummy logic with your own business logic
// for generating patient id, treatment arm, treatment assignment.
Random generator = new Random();

int patientId = Math.abs(generator.nextInt());
// perfrom randomization based on treating site

if (checklist.odm.fileOid.indexOf("B-42") >= 0) {
  openRegistration.setTreatmentAssignment("Blinded");
} else {
  if ((patientId % 2) > 0) {
    openRegistration.setTreatmentAssignment("A");
  } else {
    openRegistration.setTreatmentAssignment("B");
  }
}

// set the return status
```

```
          response.setResponseCode(ZAppBlock.PROCESSED);
          response.setResponseText(" Request processed successfully");
           response.setResponseDetailText("No details");
          openRegistration.setStatus(ZAppBlock.SUCCESS);
          openRegistration.setStatusText(" Registration complete");
          // this is to give further information on what
          // is required to complete the process, or further
          // instruction. This can be use to transmit any data.
          openRegistration
              .setSiteInstructions("Please send form to the group address.");
          openRegistration.setEligibility("Eligible");
          openRegistration.setIneligibilityReason("None");
          openRegistration.setPatientId(new Integer(patientId).toString());

          // read out from KAppConfig to find out if need to
          // persist the data to a predefined RandoNode schema.

          if (OdmUtil.getConfig("PersistData", "false").trim()
              .equalsIgnoreCase("true")) {
            this.checklist.metaDataUtil.persist();
            this.checklist.persist();
            RandoNodePersistInterface persistInterface = new RandoNodePersistInterface();
            OpenRegistrations openRegistrations = new OpenRegistrations();
            openRegistrations.setSelf(registrationResponse.getOpenResponse(),
                openRegistration);
            OpenResponses openResponses = new OpenResponses();
            openResponses.setSelf(registrationResponse.getOpenResponse());
            persistInterface.persistInterface(openRegistrations);
            // persistInterface.persistInterface(openResponses);
            persistInterface.persistInterface(openResponses.getOpenTxHeader());
          }
        }
        else{
          // there are error in the data, return either
          // Ineligible - terminate this registration
          // Incomplete - give reason why the registration does not meet
          //              validation requirement, and allow to resubmit
          openRegistration.setEligibility("Incomplete");
          response.setResponseCode(ZAppBlock.PROCESSED);
          openRegistration.setStatus(ZAppBlock.FAILURE);
        }
        successCode = true;
      }
      else{
        // tell open that meta data is missing
        response.setResponseCode(ZAppBlock.EXCEPTION);
        response.setResponseData("Missing meta data file " +
     checklist.metaDataUtil.getMetaFileName());
         // Please download the meta data file using OpenDataService, you can
        // invoke the service with the meta data file name
      }
    }
    return successCode;
  }
```

The output from the example-1 is shown below.

```
SubjectData
   DB id of the registration  = 305
   Patient id of the registration (for 2nd step and above)  = 305
   CTEP ID of the enrolling institution  = LOC.FL035
   Name of the enrolling institution  = Cedars Medical Center
   CTEP ID of the enrolling investigator = 10124
   Phone no of the enrolling investigator = (319) 272-2700
```

Example-2

This example shows the extraction the clinical data including the form name, question name and responses.

```
***** COPY the pink portion from the example-1 here*****

      /**
        * The following 2 lines of code are required from the randonode
        * when OPEN portal invokes the randonode and
        * pass in the clinical data as a String.
        * The fist line creates the ClinicalDataUtil object to do the xml extraction
        */

      ClinicalDataUtil util = new ClinicalDataUtil();

      /**
        * This call is to extract the clinical data and put the value
        * into the classes as defined. Please see each class file for
        * their usage. At this time, the metadata is also loaded into the
        * respective metadata classes from the metadata file.
        */
      util.loadClinicalData(xmlContent.toString());

      /**
        * The clinical xml file has an attribute under ODM
        * element that specify the metadata file associated
        * with this clinical responses
        */
      String metaDataFileName = util.getMetaDataFileName();
      System.out.println("MetaDataFileName = " + metaDataFileName);
      /**
        * The clinical xml file has an attribute under ODM
        * element that specify the metadata file associated
        * with this clinical responses
        */
      ClinicalData clinicalData = util.getClinicalData();
      String studyOID = clinicalData.studyOid;
      String mdvOID = clinicalData.metaDataVersionOid;
      MetaDataUtil metaUtil = util.getMetaDataUtil();
      MetaDataVersion metaDataVersion = metaUtil.getMetaDataVersion(mdvOID);
      Study study = metaDataVersion.study;
      //System.out.println(metaDataVersion);
      String studyName = study.name;
      System.out.println("Protocol number= " + studyName);
```

```
System.out.println("");


/**
  * Get the study event (clinical and metadata) details.
  * The current API handles only one Studyevent per study and one form
  * within a study event.
  */
StudyEventData studyEventData = util.getStudyEventData();
String studyEventOID = studyEventData.oid;
// the studyEventData contains the metadata for the study event also.
StudyEvent studyEvent = studyEventData.metaData;
String studyEventName = studyEvent.name;
String seRepeat = studyEvent.repeating;
System.out.println("  StudyEvent Name= " + studyEventName);
// To print all the studyEvenetData, uncomment the line below.
// System.out.println(studyEventData);

/**
  * Get the detail of the form (clinical and metadata)
  */
FormData formData = util.getFormData();
String formOID = formData.oid;
// the formData contains the metadata for the form also.
// the form metadata can also be obtained using the metaDataUtil
// object created by the ClinicalDataUtil object.
// example: Form form = util.getMetaDataUtil().getForm();
Form form = formData.metaData;
String formName = form.name;
System.out.println("    Form Name= " + formName);
/**
  * Extracting itemGroupData from formData
  */
HashMap<String, ArrayList<ItemGroupData>> itemGroupDataMap = formData.itemGroupDataMap;
Set igKeys = itemGroupDataMap.keySet();
Iterator<String> igKeysIter = igKeys.iterator();
// loop through the Items groups within a form
while (igKeysIter.hasNext()) {
  String itemGroupOID = igKeysIter.next();
  ArrayList<ItemGroupData> itemGroupDataList = itemGroupDataMap.get(itemGroupOID);
  for(int i = 0; i < itemGroupDataList.size(); i++){
    ItemGroupData itemGroupData = itemGroupDataList.get(i);
    //System.out.println(itemGroupData);
    // the itemGroupData contains the metadata for the itemGroup also.
    ItemGroup itemGroup = itemGroupData.metaData;
    String itemGrouprepeatsn = itemGroupData.itemGroupRepeatKey;
    String itemGroupName = itemGroup.name;
    System.out.println("      Item Group Name[repeat_sn]= " + itemGroupName +
                    "[" + itemGrouprepeatsn + "]-"+itemGroupOID);
    /**
     * Extracting itemData from itemGroupData
     */
    HashMap itemDataMap = itemGroupData.itemDataMap;
    Set itemKeys = itemDataMap.keySet();
    Iterator<String> itemKeysIter = itemKeys.iterator();
    // loop through the Items within a itemGroups
```

```
          while (itemKeysIter.hasNext()) {
            String itemOID = itemKeysIter.next();
            ArrayList<ItemData> itemDataList = (ArrayList<ItemData>)itemDataMap.get(itemOID);
            for(int k = 0; k < itemDataList.size(); k++){
              ItemData itemData = itemDataList.get(k);
              // the itemData contains the metadata for the item also.
              Item item = itemData.metaData;
              String value = itemData.value;
              String itemCDEID = item.cdePublicId;
              String itemName = item.name;
              String itemDataType = item.dataType;
              System.out.println("        "+itemName + "= " + value);
              // To print all the item data, uncomment the line below.
              // System.out.println(itemData.toString());
            }
          }
        }
      }
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

The output from the example is shown below.

```
MetaDataFileName = E1505_2555093_1_0_meta.xml
Protocol number= E1505

  StudyEvent Name= Registration
    Form Name= E1505 Eligibility Checklist
      Item Group Name[repeat_sn]= Eligibility[1]-IG.12
        Patient Gender= FEMALE
        What is the pateint's stage of disease= Stage II
        Fax number of the contact person responsible for sample submission =
2341234565
        E-mail address of the contact person responsible for sample submission =
tyut@yahoo.com
        Histology= Squamous cell carcinoma
        Has HIPAA authorization been obtained= Yes
        Pt Initials= LY
        Has the eligibility checklist been completed= Yes
        Phone number of the contact person responsible for sample submission=
3221342456
        Have you obtained the patient's consent for his or her blood to be kept for
use in research to learn about, prevent, treat, or cure cancer?= Yes
        In the opinion of the investigator, is the patient eligible?= Yes
        HIPAA Authorization Date= 20070507
        Have you obtained the patient's consent for his or her doctor (or somone from
the Eastern Cooperative Oncology Group) to contact him or her in the future to ask him
or her to take part in more research?= Yes
        Have you obtained the patient's consent for his or her tissue and blood to be
kept for use in future research about other health problems (for example: causes of
diabetes, Alzheimer's disease, and heart disease)?= Yes
        Has written informed consent been obtained= Yes
        Planned Chemotherapy Regimen= Cisplatin and Gemcitabine
        Date Informed Consent Signed= 20050823
        Have you obtained the patient's consent for his or her tissue to be kept for
use in research to learn about, prevent, treat, or cure cancer?= Yes
      Item Group Name[repeat_sn]= Patient Demographics / Pre-Treatment
Characteristics[1]-IG.13
        Attending MD Name= John Doe
        Patient Hospital No.= FL234
        Date of Birth= 20010508
        Patient Social Security Number= 125524587
        Registrar= Jane ren
        Country of Residence= US
        Patient Zip Code= 12548
        Method of payment= MEDICAID AND MEDICARE
        Patient Race= Black or African American
        Date= 20070514
        Patient's Sex= F
Process exited with exit code 0.
```

Example-3

The example-3 shows how to construct the response that is to be sent to OPEN portal when a clinical data message is received.

```
***** COPY the pink portion from the example-1 here*****

      /**
        * The following 2 lines of code are required from the randonode
        * when OPEN portal invokes the randonode and
        * pass in the clinical data as a String.
        * The fist line creates the ClinicalDataUtil object to do the xml extraction
        */
```

```
ClinicalDataUtil util = new ClinicalDataUtil();

/**
  * This call is to extract the clinical data and put the value
  * into the classes as defined. Please see each class file for
  * their usage. At this time, the metadata is also loaded into the
  * respective metadata classes from the metadata file.
  */
util.loadClinicalData(xmlContent.toString());

/**
  * The clinical xml file has an attribute under ClinicalData
  * element that specify the metadata file associated
  * with this clinical responses. The MetaDataVersionOID attribute
  * contains the metadata file name
  */
ClinicalData clinicalData = util.getClinicalData();
String studyOID = clinicalData.studyOid;

/**
 * Extract the clinical data(form responses) and perform the randomization
 * Please see Example2.java on how to extract the form responses.
 */


/**
 * Please perform your randomization logic and fill in the group
 * response object to be sent back to OPEN portal
 */

/**
 * Get the group reponse object partially filled in by the ClinicalDataUtil
 * class.
 */
GroupResponse groupResponse = util.getGroupResponse();

/**
 * Fill in the GroupId (your GroupId) and the GroupUrl (your
 * GroupUrl) in the MessageHeader object
 */
groupResponse.msgHeader.groupId = "ECOG";
groupResponse.msgHeader.groupUrl = "www.ecog.org/RandoNode";
/**
 * Fill in the MessagePayload object which include the
 * patient id, treatment arm, site instructions, patient eligibility,
 * and ineligibility reason where applicable from the randomization
 * result
 */
groupResponse.msgPayload.patientId = "123";
groupResponse.msgPayload.treatmentArm = "A";
groupResponse.msgPayload.ineligibilityReason = "none";
groupResponse.msgPayload.patientEligibility = "eligible";
groupResponse.msgPayload.siteInstructions = "Specimen should be shipped to ECOG.";
/**
 * Call the group response object method "toXML" and return the
 * result string to OPEN PORTAL
```

```
        */
       String xmlResponse = groupResponse.toXML();
       System.out.println(xmlResponse);

       /**
        * return the above string to OPEN Portal
        */

     } catch (Exception e) {
       e.printStackTrace();
     }
   }
}
```

The output from the example is shown below.

```
<?xml version="1.0" encoding="UTF-8" ?>
<GROUP_RESPONSE>
  <MESSAGE_HEADER>
    <MESSAGE_ID>305</MESSAGE_ID>
    <CHANGE_ID></CHANGE_ID>
    <GROUP_ID>ECOG</GROUP_ID>
    <GROUP_URL>www.ecog.org/RandoNode</GROUP_URL>
    <RECORD_TYPE></RECORD_TYPE>
    <TRANSACTION_TYPE>Insert</TRANSACTION_TYPE>
    <RECORD_ID></RECORD_ID>
    <ESYS_DB_ID></ESYS_DB_ID>
    <TRANSACTION_USER> Ravi, Rajaram</TRANSACTION_USER>
    <TRANSACTION_DATE>2007-05-22T10:51:37</TRANSACTION_DATE>
  </MESSAGE_HEADER>
  <MESSAGE_PAYLOAD>
    <PATIENT_ID>123</PATIENT_ID>
    <TREATMENT_ARM>A</TREATMENT_ARM>
    <SITE_INSTRUCTIONS>Specimen should be shipped to ECOG.</SITE_INSTRUCTIONS>
    <PATIENT_ELIGIBILITY>eligible</PATIENT_ELIGIBILITY>
    <INELIGIBILITY_REASON>none</INELIGIBILITY_REASON>
  </MESSAGE_PAYLOAD>
</GROUP_RESPONSE>
Process exited with exit code 0
```

Example-4

This example shows how to extract the metadata from the metadata xml file. This will be useful when setting up the protocol form version for the first time.

```
import com.westat.ctsu.open.node.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Set;
/**
 * Sample program to show how to use Meta Data Utility and
 * Classes to retrieve metadata
 */
public class Example4 {
```

```
public static void main(String[] args) {
    /**
     * example program to show how to use the provided metadata classes
     * to retrieve the metadata
     */
    try {
        if(args.length < 1){
            System.out.println("usage: MetaDataExample MetaDataFile");
            System.exit(0);
        }

        /**
         * this is an example code of how to use the metadata classes.
         * The metadata xml file is created by OPEN. It conforms to
         * the Cdisc standard to translate the clinical form content
         * in xml format
         */
        MetaDataUtil util = new MetaDataUtil();

        /**
         * this call to extract the metadata and put the value
         * into the classes as defined. Please see each class file for
         * their usage.
         */
        util.loadMetaData(args[0]);

        /**
         * get Study metadata information
         */
        Study study = util.getStudy();
        String studyName = study.name;
        String studyDescription = study.description;
        System.out.println(" Output from Example4 - Extraction of metadata for " +
  studyName);
          //System.out.println(util.getStudy().toString());
       /**
         * get the metadata version for this study. Per ODM, a study can
         * have multiple metadata versions
         */
        HashMap<String, MetaDataVersion> metaDataVersionMap = study.metaDataVersionMap;
        Set metaDataKeys = metaDataVersionMap.keySet();
        Iterator<String> metaDataKeysIter = metaDataKeys.iterator();
        // loop through the metadata version within the file
        while (metaDataKeysIter.hasNext()) {
            String metaDataOID = metaDataKeysIter.next();
            MetaDataVersion metaDataVersion = metaDataVersionMap.get(metaDataOID);
            //System.out.println(metaDataVersion.toString());
            // metaDataOID stors the filename of the metadata.xml
             System.out.println(" metadada file name = "+ metaDataOID);

            /**
             * the metaDataVersion has name, oid and reference to the study and
             * study event
             */
            HashMap<String, StudyEvent> studyEventMap = metaDataVersion.studyEventMap;
```

```
            /**
             * get the study event metadata information. The metadata conveys
             * if the event can be repeated within the study, the event category
             * such as Unscheduled, and whether the event is mandatory.
             */

            Set studyEventKeys = studyEventMap.keySet();
            Iterator<String> studyEventKeysIter = studyEventKeys.iterator();
            // loop through the metadata version within the file
            while (studyEventKeysIter.hasNext()) {
                String studyEventOID = studyEventKeysIter.next();
                StudyEvent studyEvent = studyEventMap.get(studyEventOID);
                // this is the name of the study event, e.g. Registration
                String studyEventName = studyEvent.name;
                System.out.println("   Study Event name = " + studyEventName);
               //System.out.println(studyEvent.toString());
            }
        }
            /**
             * the form object contains the form related metadata and
             * an array list of all the item groups within the form
             */
        HashMap formMap = util.getForm();
        Set keys = formMap.keySet();
        Iterator<String> keysIter = keys.iterator();
        while(keysIter.hasNext()){
            String oid = keysIter.next();
            Form form = (Form)formMap.get(oid);
            String formName=form.name;
            System.out.println("     Form Name = " + formName);
            //System.out.println(form.toString());


            /**
             * loop through the form to get all the itemgroups and
             * items within the itemgroup
             */
            HashMap<String, ItemGroup> itemGroupMap = form.itemGroupMap;
            Set igKeys = itemGroupMap.keySet();
            Iterator<String> igKeysIter = igKeys.iterator();
          // loop through the itemgroups within a form
           while(igKeysIter.hasNext()){
                ItemGroup ig = itemGroupMap.get(igKeysIter.next());
                String itemGroupName = ig.name;
                String igrepeatyn =ig.repeating;
                System.out.println("       Item Group Name[RepeatYN] = " +
itemGroupName+"["+igrepeatyn+"]");
                HashMap<String, Item> itemMap = ig.itemMap;
                Set itemKeys = itemMap.keySet();
                Iterator<String> itemKeysIter = itemKeys.iterator();
                // loop through the items within an itemgroup
                while (itemKeysIter.hasNext()) {
                    String itemOID = itemKeysIter.next();
                    Item item = itemMap.get(itemOID);
                    String itemName = item.name;
                    String itemCDEID =item.cdePublicId;
```

```
                        String itemDataType = item.dataType;
                        //System.out.println("         Item Name[CDEID][Datatype] = " +
    itemName+"["+ itemCDEID+"]"+"["+ itemDataType+"]");
                        System.out.println("         " + itemName+"["+ itemCDEID+"]"+"["+
    itemDataType+"]");
                        HashMap<String, CodeList> codeListMap = item.codeListMap;
                        Set codeListKeys = codeListMap.keySet();
                        Iterator<String> codeListIter = codeListKeys.iterator();
                        // loop through the codelist values for an item, if exists
                        while(codeListIter.hasNext()){
                            String codeListOid = codeListIter.next();
                            CodeList codeList = codeListMap.get(codeListOid);
                            ArrayList<String> codeListValues = codeList.codedValueList;
                            System.out.println("         Valid values for item ");
                            for (int i=0; i<codeListValues.size(); i++) {
                              System.out.println("           "+(String)codeListValues.get(i)) ;
                            }
                        }
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

The output from the example is shown below.

```
Output from Example4 - Extraction of metadata for E1505
 metadada file name = v.E1505_2555093_1_0_meta.xml
   Study Event name = Registration
     Form Name = E1505 Eligibility Checklist
       Item Group Name[RepeatYN] = Eligibility[No]
         What is the pateint's stage of disease[2004255][text]
         Valid values for item
           Stage IB
           Stage II
           Stage IIIA-N2
           Stage IIIA-T3N1
         Patient Gender[62][text]
         Valid values for item
           FEMALE
           MALE
         E-mail address of the contact person responsible for sample submission
[2597591][text]
         Fax number of the contact person responsible for sample submission
[2597590][text]
         Histology[2466][text]
         Valid values for item
           Squamous cell carcinoma
           Other Non-Small Cell Lung Cancer
         Pt Initials[2001039][text]
         Has the eligibility checklist been completed[2597470][text]
         Valid values for item
           No
           Yes
         Has HIPAA authorization been obtained[2598189][text]
         Valid values for item
           Yes
           No
           Exempt
         Phone number of the contact person responsible for sample
submission[2597589][text]
         Have you obtained the patient's consent for his or her blood to be kept for
use in research to learn about, prevent, treat, or cure cancer?[2598139][text]
         Valid values for item
           N/A - Institution outside the USA or Canada
           No
           Unknown
           Yes
         In the opinion of the investigator, is the patient eligible?[1235][text]
         Valid values for item
           No
           Yes
         HIPAA Authorization Date[2006960][date]
         Have you obtained the patient's consent for his or her doctor (or somone from
the Eastern Cooperative Oncology Group) to contact him or her in the future to ask him
or her to take part in more research?[2177934][text]
         Valid values for item
           No
           Unknown
           Yes
         Name of the contact person responsible for sample submission[2597588][text]
         Have you obtained the patient's consent for his or her tissue and blood to be
kept for use in future research about other health problems (for example: causes of
diabetes, Alzheimer's disease, and heart disease)?[58325][text]
         Valid values for item
           No
           Unknown
           Yes
         Has written informed consent been obtained[2004073][text]
         Valid values for item
           No
           Yes
         Planned Chemotherapy Regimen[2007212][text]
```

```
          Valid values for item
            Cisplatin and Docetaxel
            Cisplatin and Gemcitabine
            Cisplatin and Vinorelbine
          Please indicate the CTEP ID of the institution and Cooperative Group
affiliation of the surgeon[2597574][text]
          Date Informed Consent Signed[656][date]
          Have you obtained the patient's consent for his or her tissue to be kept for
use in research to learn about, prevent, treat, or cure cancer?[2428316][text]
          Valid values for item
            No
            Unknown
            Yes
        Item Group Name[RepeatYN] = Patient Demographics / Pre-Treatment
Characteristics[No]
          Attending MD Name[2008248][text]
          Date of Birth[793][date]
          Patient Hospital No.[905][text]
          Registrar[2172][text]
          Date[2598180][date]
          Patient Social Security Number[780][integer]
          Comments[797][text]
          Signature of treating physician[2598179][text]
          Patient Zip Code[316][text]
          Country of Residence[315][text]
          Method of payment[2614937][text]
          Valid values for item
            Medicaid
            Medicaid and Medicare
            Medicare
            Medicare and Private Insurance
            Military or Veterans Sponsored, Not Otherwise Specified (NOS)
            Military Sponsored (including CHAMPUS & TRICARE)
            National Health Service
            No means of payment (no insurance)
            Other
            Private Insurance
            Self Pay (No Insurance)
            Unknown
            Veterans Sponsored
          Patient Race[106][text]
          Valid values for item
            White
            Black or African American
            Asian
            Not Reported
            Native Hawaiian or other Pacific Islander
            American Indian or Alaska Native
            Unknown
          Patient's Sex[2413190][text]
          Valid values for item
            F
            M
Process exited with exit code 0
```

Examples-.Net

The .Net example files for all the four examples above, are available in the RandoNodeExample.msi file that can be downloaded from www.ctsu.org/open.