Cancer Trials Support Unit

CTSU – A Service of the National Cancer Institute

OPEN RandoNode Request Processing Interface (Updates for RandoNode Version 1.1)

Revision 9

20 May 2009

Document Information

Revision Information for the **Cancer Trials Support Unit** - **OPEN RandoNode Request Processing Interface (Updates for RandoNode Version 1.1)**
Document No**. OPEN/DOC## Rev. 01**

Revision History			
#	**Date**	**By**	**Description**
01	04-03-2009	Jayan Nair & Sushmita De	Initial version
02	04-07-2009	Jayan Nair & Sushmita De	Revised version after separating the workflow document and the implementation document
03	04-09-2009	Jayan Nair	Revised version after incorporating Ravi's comments
04	04-13-2009	Jayan Nair	Revised version
05	04-23-2009	Sushmita De	Incorporated the structural changes to user class and method changes in ClinicalDataUtil class.
06	04-24-2009	Jayan Nair	Changed patientStatus field from optional to mandatory.
07	05-01-2009	Sushmita De	Introduced "How to Install/Upgrade To RandoNode 1.1.0.0 Build" section.
08	05-08-2009	Jayan Nair	Changed the case of Method of payment valid values to match the caDSR specifications – They use upper case for all values except for Unknown.
09	05-20-2009	Jayan Nair	Screenshots updated to reflect the latest OPEN screens
Last Saved By **Jayan Nair** on **5/20/2009 11:13 AM**			
File Location: **C:\CTSU\Sandbox\apps\OPEN\docs\RandoNode\RandoNode_API_1.1.doc** VSS Location: **$/Sandbox/apps/OPEN/docs/RandoNode/RandoNode_API\RandoNode_API_1.1**			

This document was prepared for:

This document was prepared by:

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

Jayan Nair 301-212-2126 & Sushmita De 301-738-8311

Table of Contents

Tables

Figures

References

#	Document	Location	Description
1.	OPEN RandoNode Request Processing Interface (RandoNode_API_1.0)	https://www.ctsu.org/open/default.asp?fName=open/Randonode/docs	Original RandoNode Request Processing interface document
2.	NCI Standard Demography Template	https://gforge.nci.nih.gov/docman/view.php/401/8749/Demography%20Template%20-%20FINAL%2020071009.pdf	NCI Specification for Standard Demography Forms
3.	Instructions and Guidelines -Clinical Data Update System (CDUS)	http://ctep.cancer.gov/protocolDevelopment/electronic_applications/docs/cdus_ig_3r4.pdf	
4.	CDE Browser	https://cdebrowser.nci.nih.gov/CDEBrowser/	To find additional information related to the CDE Public IDs mentioned in this document

1. Introduction

1.1. Overview

This document is a supplement to the original "OPEN RandoNode Request Processing Interface document" [Reference #1] which described the interface and communication protocols between OPEN portal and the RandoNode.

The objective of this document is to describe the changes brought in by RandoNode version 1.1 on the interface objects after the introduction of the following features.

1. **Standard Demography Form:** With OPEN portal version 1.1, a new demography form was introduced in OPEN to separate the demography data from that of the Eligibility Checklist (EC) data. This was done due to the following reasons.
    a. To comply with caBIG standards to use the NCI Standard Demography Module uniformly across all the protocols.
    b. To support advanced features such as pre-filling of demography data as well as existing patient verification as described below.
2. **Pre-fill Demography Data:** Pre-fill the demographic data from the group's database, based on the unique patient ID assigned to a patient. This process happens before filling in the demography data.
3. **Matching Existing Patient Registration:** Verify a previous registration in the same study (duplicate registration) or existing registration in a different study when a patient is registered to a protocol. This verification happens after filling in the demography data.
4. **Dynamic Version Adaptation:** To facilitate OPEN to communicate with different versions of the RandoNode a dynamic version adaptation feature is added to the RandoNode framework.
5. **User Class Structure Modification:** To support multiple OPENUserType for an user, existing User class structure is modified.

Only the changes brought in by the above features are described in this document. For more detailed description on the RandoNode request processing interface, the reader is requested to refer to the original document [References #1].

1.2. Document Organization

This document is developed in two parts. In Part I of this document, the details of workflow changes brought in by these changes are described using use cases and workflow diagrams.

In Part II of the document specific technical details related to the interface changes introduced by these enhancements are described.

Since Part II of this document builds on the concepts introduced in Part I of the document, developers are requested to review Part I of this document before reviewing the implementation details in Part II.

In Part III of the document User class structure change details and few related utility methods are described.

1.3. Target Audience

Part I of this document is intended for general audience at the cooperative groups involved with OPEN project.

Part II and Part III of the document is intended for the development team at the cooperative groups implementing and extending the RandoNode for processing patient enrollments under the OPEN portal.

Part I: General Workflow Changes

2. Pre-Fill the Demography Form

2.1. Requirements

While registering a new patient to a protocol, OPEN should provide facilities to pre-fill the demography form for an existing patient, using the previously assigned patient ID.

This feature should be configurable on a per group basis. If a group doesn't have the ability to provide the demography data based on the existing patient ID, no communication will be made to the group for fetching the demography information.

2.2. Use Case

When registering a patient, if the patient ID is available, then the site registrar will enter the patient ID in OPEN.

The registrar will then click a button **Populate Data** to get the information from the cooperative group as shown below:

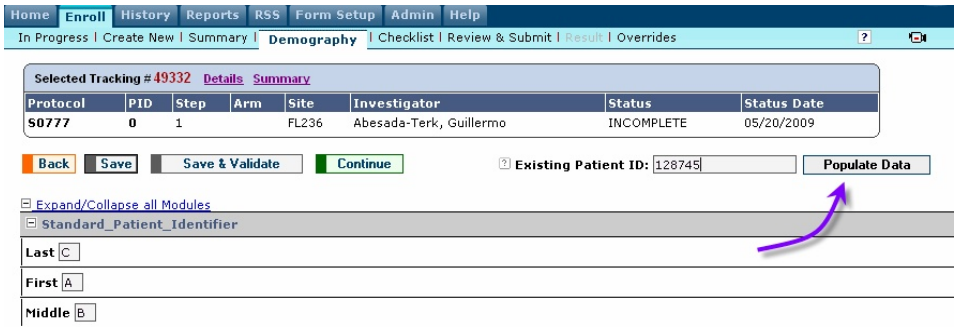

Figure 1: Pre-filling of demography data

When the user clicks the Populate Data button, OPEN will submit this request to the group's RandoNode.

The RandoNode will try to find the information corresponding to that patient in its database and will return the patient information or an indicator that the patient is not found.

If OPEN receives the patient information from the group, that information will be pre-populated on the demography form.

If the patient was not found then an appropriate error message will be displayed on the screen.

The registrar can verify the pre-filled information to match with what they have in hand. If the information doesn't match then they need to verify the accuracy of the existing patient ID.

Registrars should be able to edit the pre-filled information such as Method of Payment. Input from the groups is required on what other demography information the registrars will be allowed to edit.

If the patient's information was not found in the group's database, then the user has two choices.

Verify the entered existing patient ID for accuracy and correct any errors, OR

If the existing patient ID cannot be matched then they can register the patient as a new patient.

Once the demography form is reviewed or edited, the registrar can continue to the next step by clicking the **NEXT** button.

2.3. Workflow for Pre-Filling the Demography Data

The following workflow diagram, explains this process.

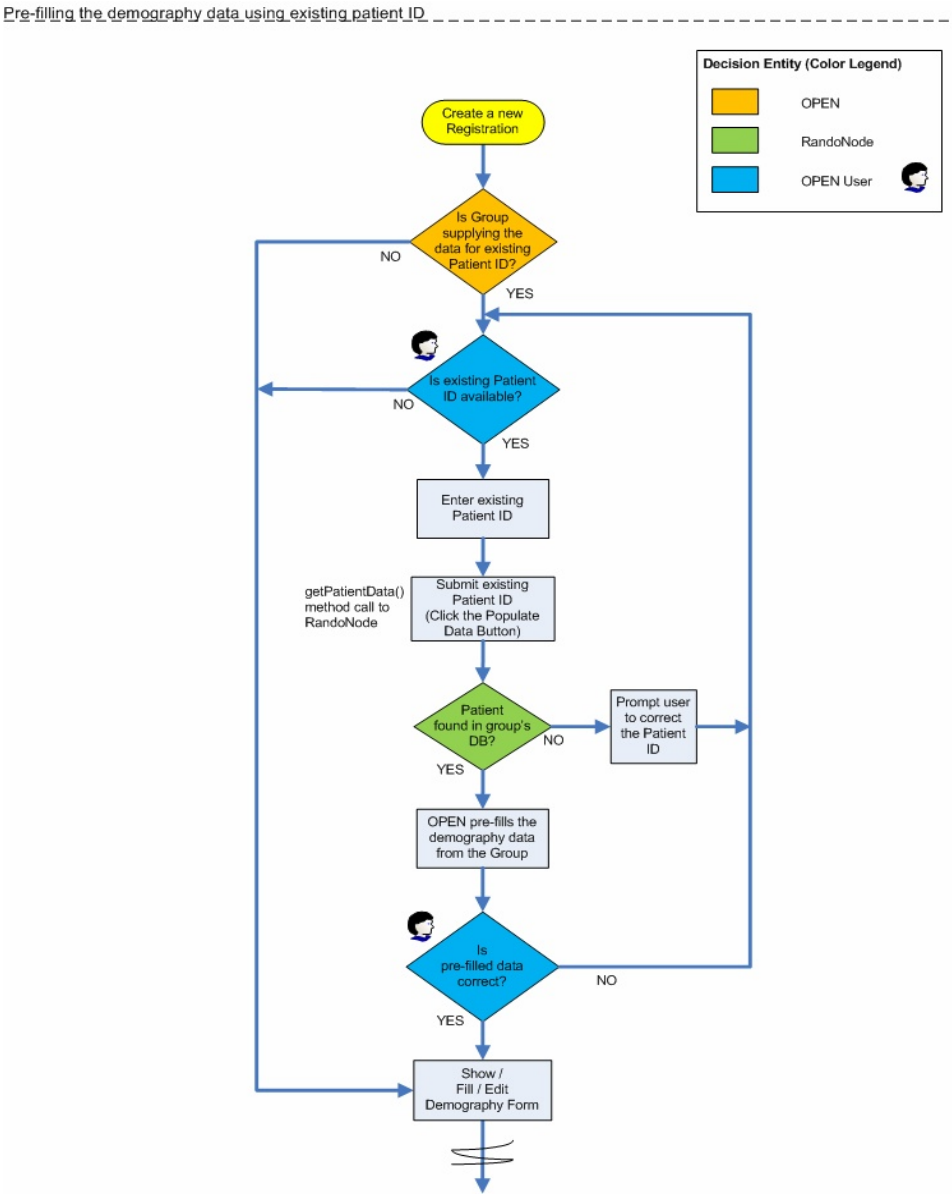

Figure 2: Workflow diagram to pre-fill demography data

3. Previous Patient Verification

3.1. Requirements

OPEN should provide a facility for the groups to verify if the same patient was already registered to this protocol.

OPEN should provide a facility for the groups to verify if this patient has been registered to a different study by the same group so as to reuse the patient ID already assigned to a patient.

This feature should be configurable on a per group basis. If a group doesn't have the ability to verify this information, no communication will be sent to the group for this feature.

3.2. Use Case

The registrar creates a new registration for a patient in OPEN by selecting the protocol, site and the investigator.

Next, the registrar goes to the demography screen and fills in the demography data for the patient.

The registrar then clicks the **NEXT** button on the demography screen.

At this time, OPEN will communicate with the corresponding group RandoNode and supply the demography data filled-in by the registrar. Only those groups who have the ability to verify duplicate patients in this study (or existing patient in another study) will be contacted. Otherwise OPEN will take the registrar to the Eligibility checklist (EC) form.

When the RandoNode receives the demographic data, it will try to find a match with any patients in their database using the supplied demography data. If a match is found then the RandoNode will send back the information of the existing patient to OPEN. If a match is not found then the RandoNode will reply to OPEN without filling in the existing patient information.

If OPEN does not receive the existing patient information from the group RandoNode then it will proceed and show the EC form to the Registrar. In this scenario all the background communication that occurred will be transparent to the Registrar. This will be the typical scenario for most new registrations.

If the group finds the patient in the same study or different study, it will send any one of the following flags to OPEN depending on the matching scheme used by the group as well as if the match was found in this study or in another study.

```
{"PT_IS_DUPLICATE",
"PT_IN_OTHER_STUDY",
"PT_POSSIBLY_DUPLICATE",
"PT_POSSIBLY_IN_OTHER_STUDY"}
```

The matching scheme used by the RandoNode can be strict or weak. In the first two cases, the match was made using a strict matching scheme (say Social Security Number) and in the last two cases the match was made using a weak matching scheme (say first and last initial and zip code).

If OPEN receives the existing patient information filled-in by the group RandoNode, along with the flag PT_IS_DUPLICATE or PT_POSSIBLY_DUPLICATE then it will display the following choices to the user.

This patient was found to be enrolled on this date (MM/DD/YYYY) with Patient ID (xxxxxx) and was registered by (Registrar Name) in this study. Please confirm any one of the following:

Confirm that the patient is a duplicate registration. (PT_IS_DUPLICATE). Do not do a new registration.

Confirm that this patient is a new registration (PT_CONFIRMED_NEW). The match between this patient

and the other patient is just a coincidence. This message is shown only if OPEN received PT_POSSIBLY_DUPLICATE flag from the RandoNode since the matching was weak.

Keep this registration pending and will revisit at a later time for further review. (REVIEW_LATER)

If OPEN receives the flag PT_IN_OTHER_STUDY or PT_POSSIBLY_IN_OTHER_STUDY, then it will display the following choices to the user.

This patient was found to be enrolled on this date (MM/DD/YYYY) with Patient ID (xxxxxx) and was registered by (Registrar Name) in the following study (study name, step) and credentialing (Crediting Investigator, Crediting Site). Please confirm any one of the following:

Confirm that the patient is the same patient registered in the other study (PT_SAME_AS_EXISTING_PT).

Confirm that this patient is a new patient (PT_CONFIRMED_NEW). The match between this patient and the other patient is just a coincidence. This message is shown only if OPEN received PT_POSSIBLY_IN_OTHER_STUDY message from the RandoNode since the matching was a weak matching.

Keep this registration pending and will revisit at a later time for further review. (REVIEW_LATER)

In scenarios 8 and 9, if the user selection was PT_CONFIRMED_NEW, then registration will proceed by continuing to the EC form.

If the choice was PT_SAME_AS_EXISTING_PT, then OPEN will persist this Existing Patient ID in the database and will continue to the EC from. This Patient ID will be sent to the group when registration data is finally submitted for Randomization.

In cases 10 and 11 a new flag will be sent to the RandoNode indicating the above status, so that further verification can be avoided when submitting the EC form. This flag will be cleared if the user modifies the demography form before submitting the EC form.

If the choice was PT_IS_DUPLICATE, then the registration will be abandoned (VOID state)

If the choice was REVIEW_LATER, then the registration will be marked as INCOMPLETE and will be available for review later. If user tries to resubmit the registration which was in INCOMPLETE state, the demography data will be validated again using the existing patient check.

After the user made the selection of PT_CONFIRMED_NEW or PT_SAME_AS_EXISTING_PT, there is no further existing patient verification done at the RandoNode, unless the demography data was changed by the user. In such cases the value of the above flag will be reset to PT_NOT_VALIDATED. This will trigger the existing patient verification at the RandoNode whenever this data is submitted.

3.3. Workflow for Previous Patient Verification

The following workflow diagram, explains the above use case description.

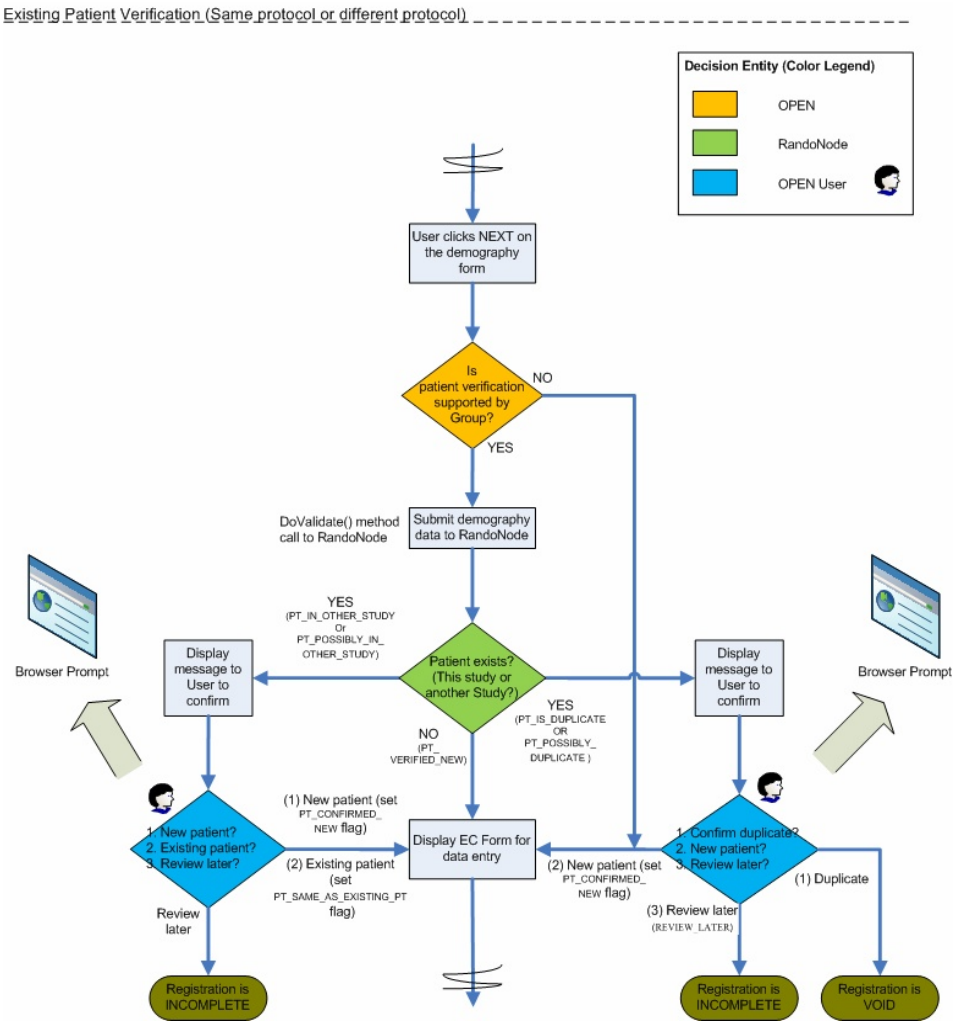

Figure 3: Workflow diagram to verify previous patient registrations

Part II – Interface Implementation Details

4. Summary of Changes

The following are the summary of changes for RandoNode version 1.1.

4.1. New Web Methods

Two new web methods are added to the RandoNode webservice as described below.

4.1.1. getVersion()

This method is used to get the version of the RandoNode so that OPEN can send the appropriate version of objects to the RandoNode. Please see section 5.1 for more information on this web method.

4.1.2. getPatientData()

This method is used to pre-fill the demography data when an existing patient ID is available. OPEN invokes this method with an existing patient ID. If the cooperative group has the data for this patient, the RandoNode will send back the demography data. Please see section 5.1 for more information on this web method.

4.2. New Interface Classes

Two new interface classes (***Demography*** and ***ExistingPatient***) are added as member attributes of the ***RegistrationResponse*** object as described below.

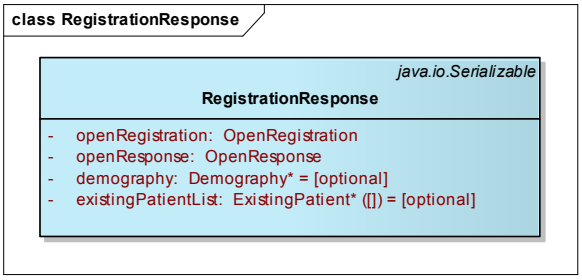

Figure 4 : New object attributes added to RegistrationResponse class

4.2.1. Demography

This class was added to send the demography data from the RandoNode to OPEN in response to OPEN's ***getPatientData()*** method request. When making a ***getPatientData()*** call, OPEN will provide the existing patient ID as an input. Please see section 5.2.1 for further information on the ***Demography*** class.

This is an optional attribute and should be filled in only in the case of a ***getPatientData()*** method call.

4.2.2. ExistingPatient

This class is useful to verify whether a patient already exists in a different study or within the same study. During a new enrollment, if a match of the demography data is found between the new enrollment and an existing patient, then the RandoNode can send the existing patient data to OPEN through this object. Please see section 5.2.2 for further information on the ***ExistingPatient*** class.

This is an optional attribute of the ***RegistrationResponse*** class and should be filled-in by the RandoNode

only in two cases. In ***doValidate()*** call to validate the demography data or in ***doRegister()*** call. The ***operation*** flag of ***OpenRequest*** object will indicate if the ***doValidate()*** or ***doRegister()*** calls should validate the demography data alone or validate both the demography and EC data.

NOTE: If the RandoNode is doing a weak matching to compare the demography data (say for example using the last and first initial and the zip code) then it is possible to find more than one match for the ***ExistingPatient***. To accommodate this scenario, the ***ExistingPatientList*** attribute is made an array of ***ExistingPatient*** objects.

4.3. Attribute Changes to Existing Classes

Three new attributes are added to ***OpenRegistration*** class.

The first one is ***userResponse*** which will send the user response to the RandoNode in the case of pre-filling of the demography form as well as in the case of matching the demography data with an already enrolled patient.

The second attribute is ***patientStatus*** which can be used by the RandoNode to send the updated status of patient's continued participation in a study.

The third attribute is ***offStudyReason*** which will be used by the RandoNode to send the reason for a patient discontinuing the participation in a study.

Please see section 5.3.1.2 for further information on these attributes.

4.4. Changes to Existing Methods

4.4.1. doValidate()

The usage of the ***doValidate()*** method is further extended to facilitate the matching of the demography data with that of existing patients. For details of these changes, please refer to section 5.4.1 of this document.

4.4.2. doRegister()

The usage of the ***doRegister()*** method is further extended to facilitate matching the demography data with that of existing patients. This was necessary since user might have changed the demography data after the initial ***doValidate()*** check or that the user might have bypassed the demography data check.

The existing design recommendation for the ***doRegister()*** method already requires that the ***doRegister()*** method internally call the ***doValidate()*** method***.*** If the group RandoNode is following this strategy then there is no change in the ***doRgister()*** method.

For details of these changes, please refer to section 5.4.2 of this document.

5. RandoNode API - Details of Changes

5.1. New RandoNode Web Methods

5.1.1. getVersion() and getPatientData()

The ***getVersion()*** method was added for dynamic version adaptation by OPEN. This will facilitate seamless evolution and growth of the group RandoNodes through multiple interfaces of OPEN over a course of time.

This web method will allow the groups to adhere to their own timeframes in terms of feature upgrades without all the groups needing to upgrade at the same time even when the RandoNode interface of OPEN has been changed.

This method will give the flexibility for OPEN to add enhancements to its interface to support new features for specific RandoNodes, without adversely impacting the RandoNodes which have not upgraded to use the new feature.

By using the ***getVersion()*** method, OPEN can query the group RandoNodes for their version and send them the appropriate version of the objects.

The advantage of having this approach is that, groups can select their own schedules for adding various features available in OPEN to their RandoNode, without OPEN mandating any specific migration timeframe.

The interface of the two new methods (***getVersion()*** and ***getPatientData()***) added in version 1.1 are given below.

Table 1: Interface of new RandoNode web methods

Method	Parameters		Description
	Input	Output	
getVersion()	None	String value containing the specific formatted version number of the RandoNode. The version string is a four part number separated by a period in between. Eg: 1.0.1.14 [Major.Minor.Build.Revision]	• This method will return a formatted string containing the RandoNode's version. • This method will be supported by the RandoNode starter kit; hence no implementation is necessary by the groups who are using version 1.1 of the RandoNode. • **Note:** Those groups that don't use the CTSU provided RandoNode; they must provide the implementation for this method in order for them to be able to use the features of version 1.1 of OPEN and RandoNode. • With version 1.1 of OPEN, the version number is made part of the WSDL. The groups implementing the ***getVersion()*** method can return this version number to OPEN.
getPatientData()	OpenRequest OpenRegistration	RegistrationResponse	• By using this method, OPEN can call the RandoNode by providing the existing patient ID and protocol information. • The existing patient ID will be sent through the ***OpenRegistration.patientId*** attribute. • The RandoNode will return the demography data (using the ***RegistrationResponse.demography*** attribute) for that patient Id if that patient is found in the group's database. • If a patient was not found in the group's database for the supplied patient Id then the ***demography*** attribute will be set to null by the RandoNode. • The ***existingPatientList*** will remain as null due to the default initialization. • The ***getPatientData()*** method will be supported by the group RandoNodes that implement the pre-filling of demography data using existing patient Id. • The CTSU RandoNode starter kit will provide a reference implementation for this web method. • This method will be used in future to get the current registration data of a patient including ***patientStatus*** for an accrual, current site, current investigator and current demographic data. CTSU plans to use the ***getPatientData()*** method for payment purposes.

NOTE: The interface of the ***getVersion()*** method is kept simple by eliminating any object parameters. The

reason behind this decision is that when OPEN needs to get the version information from a RandoNode, OPEN might not know what version of objects to use to communicate with the RandoNode. Having a simple interface will resolve this object version ambiguity.

5.1.2. getPatientData() method – Request Flow Example

The example of a ***getPatientData()*** method call is illustrated in the following picture. OPEN portal will send the existing patient ID through the ***OpenRegistration.patientId*** attribute.

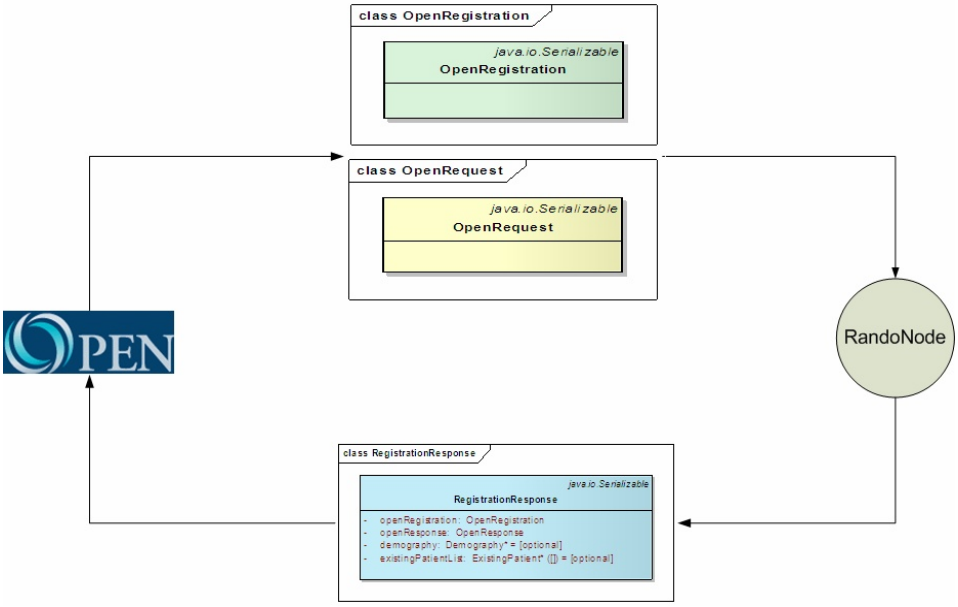

Figure 5: Object flow when invoking getPatientData() method

When the RandoNode receives the ***getPatientData()*** call, if the corresponding patient is found in group's database then the RandoNode will create a new ***Demography*** object. Fill it with the patient data and send it back to OPEN by setting it to the ***RegistrationResponse.demography*** attribute.

In future this method will also be used to get the current registration data of a patient including patient status for an accrual, current site, current investigator and current demographic data.

The ***existingPatientList*** attribute will be set to null by the RandoNode in this method call.

5.2. New Interface Classes

To support the new features, two new interface classes have been added to the RandoNode. Similarly a few new attributes are added to existing classes. These changes are shown in the following picture. The new classes and attributes are highlighted by blue squares in the picture.

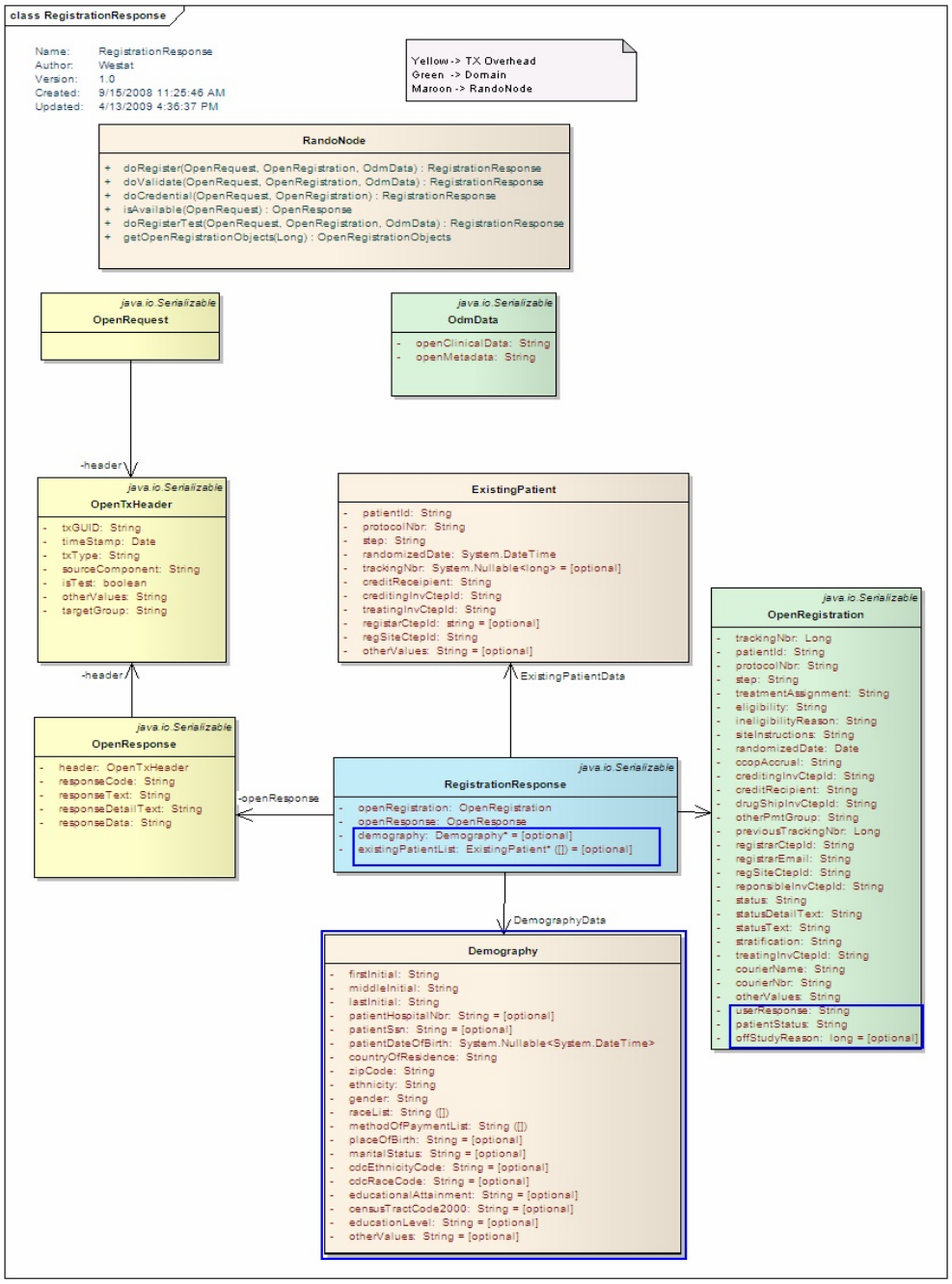

Figure 6: Changes to interface classes

The two new object members added to the ***RegistrationResponse*** class are also shown in the following code listing.

```
public class RegistrationResponse
{
    // Existing object members
    private OpenRegistration openRegistration;
    private OpenResponse openResponse;

    //-----------------------------------------------------
    // New object members added with RandoNode version 1.1
    //-----------------------------------------------------

    // [OPTIONAL]
    private Demography demography = null;

    // [OPTIONAL]
    private ExistingPatient[] existingPatientArray = null;
}
```

Figure 7: New object attributes added to RegistrationResponse class

The first is an object of class ***Demography***. This class was added to support the pre-filling of the demography data when an existing patient ID is given to the RandoNode through the ***getPatientData()*** method call. This object will be filled-in and set to the ***demography*** attribute of the ***RegistrationResponse*** object when the group receives a ***getPatientData()*** method call and the supplied patient ID matches with a patient already in the group's database. This method is called before the demography data is filled-in.

ExistingPatient class is used to send back the information of an existing patient by the RandoNode when a new enrollment's demography data matches with that of an existing patient (or patients) in the group's database. This is done when the RandoNode receives a ***doValidate()*** call to validate the demography data during a new enrollment. This method call happens after the demography data is filled-in.

The ***existingPatientList*** attribute is made as an array of ***ExistingPatient*** objects to accommodate the fact that if a weak matching is made by the RandoNode, it is possible to find more than one existing patient corresponding to the supplied demography data.

5.2.1. Demography – New Class

The Demography class consists of the attributes corresponding to all the questions proposed in NCI's Standard Demographic Template CRF. At this time, OPEN expects the groups to fill-in only those data that is available (normally those correspond to the data submitted to CDUS). The optional attributes are indicated appropriately in the class diagram.

In addition to the demography question related attributes, the Demography class also contains patient identifier attributes such as patient initials, hospital ID etc.

The attributes of ***Demography*** class and their explanations are given in the following code listing.

```
public class Demography
{
    private String lastInitial;             // CDE Public ID - 2658183
    private String firstInitial;            // CDE Public ID - 2658182
    private String middleInitial;           // CDE Public ID - 2658163
    private String patientSsn;              // CDE Public ID - 780
    private String patientHospitalNbr;      // CDE Public ID - 905
    private String ethnicity;               // CDE Public ID - 2192217
    private String gender;                  // CDE Public ID - 2200604
    private DateTime patientDateOfBirth;    // CDE Public ID - 793
    private String countryOfResidence;      // CDE Public ID - 315
    private String zipCode;                 // CDE Public ID - 2179606
    private String[] raceArray;             // CDE Public ID - 2192199 - [MAXIMUM ARRAY LENGTH = 7]
    private String[] methodOfPaymentArray;  // CDE Public ID - 2003309 - [MAXIMUM ARRAY LENGTH = 12]
    private String censusTractCode2000;     // CDE Public ID - 2681528
    private String cdcRaceCode;             // CDE Public ID - 2200286
    private String cdcEthnicityCode;        // CDE Public ID - 2200284
    private String educationLevel;          // CDE Public ID - 2674076
    private String educationalAttainment;   // CDE Public ID - 2681552
    private String maritalStatus;           // CDE Public ID - 2188083
    private String placeOfBirth;            // CDE Public ID - 2682009
}
```

Figure 8: Attributes of Demography class

For the valid values of the ***Demography*** class attributes, please refer to Table 7 of Appendix I. For the explanations of the attributes, please refer to the corresponding CDE Public ID mentioned against these attributes by accessing the CDE Browser website [References **#Error! Reference source not found.**].

Though the country of residence CDE (315) does not support enumerated country code list, OPEN will associate country code list to the valid values of the question associated with CDE 2006183.

NOTE: ***raceList*** and ***methodOfPaymentList*** are defined as collections (arrays) of strings to accommodate the fact that it is possible to have multiple races for a single patient as well as a single patient can choose multiple method of payment options.

5.2.2. ExistingPatient – New Class

The attributes of ***ExistingPatient*** class and their explanations are given below.

```
public class ExistingPatient
{
    private String protocolNbr;             // Original study number
    private String step;                    // Step for the original study
    private String patientId;               // Existing patient Id
    private DateTime randomizedDate;        // Current registration date
    private String creditReceipient;        // Current crediting group CTEP id

    // Treating investigator can be the same as the crediting Investigator
    private String treatingInvCtepId;       // Treating investigator CTEP id (If available)

    private String regSiteCtepId;           // Current institution CTEP id
    private String creditingInvCtepId;      // Current crediting investigator CTEP id
    private String registarCtepId;          // CTEP id of the orginial registrar
    private long trackingNbr = -99999999;   // Tracking number assigned by OPEN if available
}
```

Figure 9: Attributes of ExistingPatient Class

5.3. Attribute Changes to Existing Classes

5.3.1. Changes to OpenRegistration Class

5.3.1.1. Changes to OpenRegistration.status attribute

Right now the valid values of ***OpenRegistration.status*** attribute are the following:

{"SUCCESS", "FAILURE", "PENDING-GROUP"}.

With RandoNode 1.1 version, we are expanding the valid values of the status attribute as described below.

This is to include the results of the ***doValidate()*** method call to validate the demography data by matching it against the demography data of an existing patient.

Table 2: Valid values of OpenRegistration.status attribute

Attribute	Data Type	Valid value	Meaning of the value
status	v32	SUCCESS	As long as the data is processed successfully and the patient is determined to be eligible or not eligible, the status will be SUCCESS.
		FAILURE	If the data fails the validation checks, then groups should return FAILURE.
		PENDING-GROUP	If due to some reason, the patient eligibility or the requested operation such as credentialing, validation cannot be performed 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 metadata file and complete the setup.
		PT_IS_DUPLICATE	A strict matching is made by the group (for example the patient is matched using the social security number) and found that this patient was already registered in this study. Site registrars will not be able to create a new registration for this patient in this scenario. **Options for the Registrar:** Being a duplicate, the registrar can drop this registration, or they can put it pending for further review.
		PT_IN_OTHER_STUDY	A strict matching is made by the group (for example the patient is matched using the social security number) and found that this patient already exist in the group's database. However the patient's existing enrollment was on a study (or studies) different from the current study in which they are being enrolled. **Options for the Registrar:** The registrar can accept the existing patient ID and continue the registration. Or put the registration pending for further review.

Attribute	Data Type	Valid value	Meaning of the value
		PT_POSSIBLY_DUPLICATE	There was a match found between the patient's demography data and a patient already enrolled in this study. However the match was not strict (For example the match was made using first and last initials and the zip code). **Options for the Registrar:** The registrar can further scrutinize and confirm that this is actually a new patient and the match is just a coincidence. In this scenario the registrars can continue with the new registration. Or can confirm that the patient is duplicate and abandon the registration. Or they can put it pending for further review.
		PT_POSSIBLY_IN_OTHER_STUDY	There was a match found between the patient's demography data and a patient enrolled in another study. However the match was not strict (For example the match was made using first and last initials and the zip code). **Options for the Registrar:** The registrar can further scrutinize and confirm that this is actually a new patient and the match is just a coincidence. In this scenario the registrars can continue with the new registration. The registrar can do further scrutiny and accept the existing patient ID and continue the registration. Or they can put it pending for further review.
		EXISTING_PT_MISMATCH	This flag is added to handle a special case where the user after accepting an existing patient ID modified the vital demography data used in matching the patient. Then the modified data was submitted to the RandoNode. The above scenario can be detected by the RandoNode when the ***OpenRegistration.patientId*** field is filled-in, however the vital demographic information submitted doesn't match with the patient information for the submitted patient in the group's database.

OPEN will use the above flags to create the appropriate message for the user. For example in the case of receiving "PT_IS_DUPLICATE" flag from the RandoNode, the registrars will be given only two choices (1) abandon the registration or (2) keep it pending for further review.

5.3.1.2. New attributes added to OpenRegistration

Three new attributes are added to the ***OpenRegistration*** class. The first is a string attribute called ***userResponse***. This field will be used by OPEN to communicate the user's response to the RandoNode after a patient was found in the group's database.

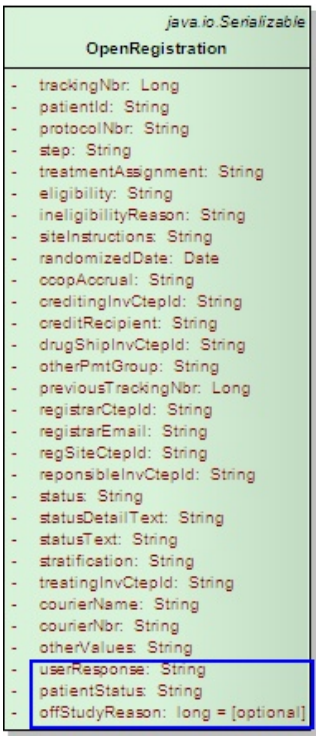

Figure 10: New attributes added to OpenRegistration class

This flag can have any one of the following string values.

Table 3: Valid values of OpenRegistration.userResponse attribute

Attribute	Data Type	Valid value	Meaning of the value
String ***userResponse***	v32	NOT_APPLICABLE	The ***getPatientData()*** and ***doCredential()*** web methods do not use this flag. Hence in those function calls; the value of this flag will be set to 'NOT_ APPLICABLE" by the OPEN portal.
		PT_SAME_AS_EXISTING_PT	This value indicates that the registrar is confirming that the match found by the RandoNode is correct. The new patient to be enrolled is same as an existing patient in the group's database. This value was added for additional clarity; even though the same information can be deciphered by the fact that the ***patientId*** of the ***OpenRegistration*** object sent by OPEN will match what is in the group's database. This value will be sent by OPEN to the RandoNode only if the ***status*** value mentioned in Table 2 is equal to PT_IN_OTHER_STUDY Or

Attribute	Data Type	Valid value	Meaning of the value
			PT_POSSIBLY_IN_OTHER_STUDY
		PT_CONFIRMED_NEW	This value means that even though the patient seems to exist in the group's database, the registrar after some scrutiny is saying that the match is just a coincidence and that the patient is indeed a new patient.
		PT_NOT_VALIDATED	This value is send to the RandoNode to indicate that the Patient's demography data is not yet validated. This can happen in the first submission of demography data to the group or in subsequent submissions if the demography data was changed after the first submission. When receiving this flag the group should do the validation as though this is the first submission of the demography data.

The second new attribute added to ***OpenRegistration*** is a string attribute called ***patientStatus***. This field will be used by the RandoNode to communicate the patient's latest status with respect to the patient's participation in the protocol.

This flag can have any one of the following string values.

Table 4: Valid values of OpenRegistration.patientStatus attribute

Attribute	Data Type	Valid value	Meaning of the value
String ***patientStatus***	v32	NOT_APPLICABLE	Default value filled in by OPEN before sending it to the RandoNode.
		PT_ON_STUDY	Patient is continuing to participate in the study.
		PT_OFF_STUDY	Patient is no longer participating in the study. If ***patientStatus*** = PT_OFF_STUDY then the ***offStudyReason*** attribute is also to be filled-in by the RandoNode with a valid value.

The third new attribute added to ***OpenRegistration*** is a numeric attribute called ***offStudyReason***. This field will be used by the RandoNode to communicate the patient's reason for discontinuing their participation in the study. OPEN will read this attribute only when the ***patientStatus*** attribute is equal to "PT_OFF_STUDY".

Please see the reference [Reference **#Error! Reference source not found.**] for more information on this topic.

This flag can have any one of the following numeric values.

Table 5: Valid values of OpenRegistration.offStudyReason attribute

Attribute	Data Type	Valid value	Meaning of the value
long offStudyReason	number	1	Protocol-defined follow-up completed
		2	Patient lost to follow-up
		3	Patient refused follow-up
		4	Death
		5	Adverse Event/Side Effects/Complications
		98	Other

5.3.2. Changes to OpenRequest.operation attribute

The ***operation*** flag of the ***OpenRequest*** object is further defined and refined for all the following scenarios.

Table 6: Valid values of OpenRequest.operation attribute

Attribute	Data Type	Valid value	Meaning of the value
String ***operation***	v32	REGISTER_PATIENT	This will be the value of the operation flag for a normal ***doRegister()*** web method call.
		ManualRegistration	This value is used in ***doRegister()*** when doing manual registration, which is a special case of the ***doRegister()*** function. This value is specified in lower case for backward compatibility. The implementation will handle both upper and lower cases.
		DataTransfer	This value is used in ***doRegister()*** when doing a data transfer of the registration data from OPEN to the RandoNode, which is a special case of the ***doRegister()*** function. This value is specified in lower case for backward compatibility. The implementation will handle both upper and lower cases.
		VALIDATE_ALL_DATA	This value is used in ***doValidate()*** function to validate all data, including Eligibility Checklist data (EC Data) and Demography Data.
		VALIDATE_DEMOGRAPHY_DATA	This value is used in ***doValidate()*** function to validate JUST the Demography Data.
		POPULATE_DEMOGRAPHY_DATA	This value is used in ***getPatientData()*** function to pre-fill the Demography Data.
		DO_CREDENTIAL	This value is used in ***doCredential()*** function.
		IS_AVAILABLE	This value is used in ***isAvailablel()*** function.

5.4. Changes to Existing Methods

5.4.1. Changes to doValidate() web method

In the existing implementation, the ***doRegister()*** method will internally call the ***doValidate()*** function. The ***doValidate()*** function is also used in validating group edit checks.

With the introduction of standard demography form we are extending the use of the ***doValidate()*** function to do the verification of the demography data also.

In this ***doValidate()*** call, the groups can compare the submitted demography data with that of the previously registered patients and see if there is a match with a previously registered patient, within this protocol or with another protocol.

There is no change in the interface of this method. However there are changes in the ***RegistrationResponse*** class and the ***OpenRegistration*** class which will change the behavior of this function.

```
public RegistrationResponse doValidate(OpenRequest openRequest,
                    OpenRegistration openRegistration, OdmData odmData);
```

After the user fills-in the demography data and clicks the NEXT button as shown in the following picture, OPEN will make a call to the ***doValidate()*** function of the RandoNode to verify the demography data of the patient.

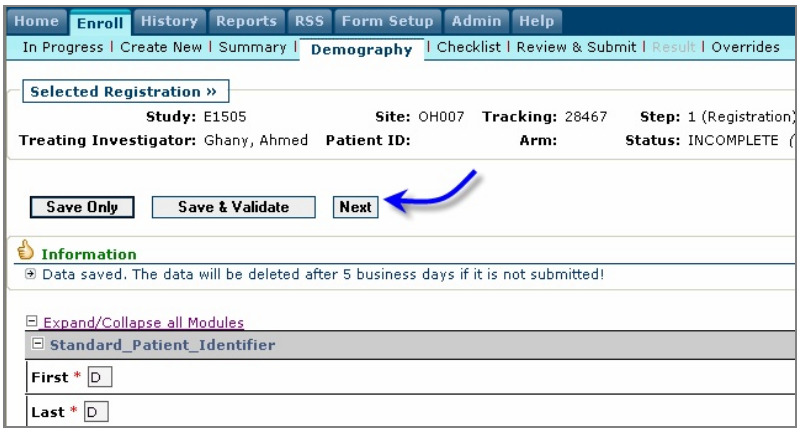

Figure 11: Triggering the doValidate() call to verify demography data

In the above scenario, OPEN will fill the ***operation*** flag of the ***OpenRequest*** object with the following value.

```
openRequest.operation = "VALIDATE_DEMOGRAPHY_DATA";
```

Using this flag the RandoNode can identify if this validation request is coming from the demography form or from the EC form.

Anytime a ***doValidate()*** call is received by the RandoNode, they must do the following three validations **IN SEQUENCE** depending on the ***operation*** flag.

Check for existing patient (in this protocol and other protocols – if the group supports this feature).

Validate demography data.

Validate EC data.

NOTE: When the ***operation*** flag is "VALIDATE_DEMOGRAPHY_DATA", then the groups should only do steps 1 and 2 above. The reason is that the EC data may not be available when the Demography Data is sent for validation.

When a RandoNode receives the ***doValidate()*** call, given below are the steps that should happen at the RandoNode.

1. Check the ***operation*** flag.
2. If (openRequest.operation == "VALIDATE_DEMOGRAPHY_DATA") then proceed with the validation as shown below.
3. If the group supports verification of previous registration of the patient then do that verification first using the following steps. If verification of previous registration is not supported, then do the demography data validation as per step 3.a.iv.2.
    a) If ***OpenRegistration.userResponse*** == "PT_NOT_VALIDATED" then:
        i. If the ***OpenRegistration.patientId*** field is filled-in then this patient already exists in the group's database. Hence verify that the user has not changed any vital demography data that was used in the existing patient check. This will be done by the RandoNode by retrieving the existing patient demography data using the supplied patient ID and comparing it with the submitted demography data. If the vital demography data was changed then return the function with ***OpenRegistration.status*** = "EXISTING_PT_MISMATCH", otherwise continue to validate the demography data as described in 3.a.iv.2. If EXISTING_PT_MISMATCH was detected by the RandoNode then the

OpenRegistration.statusText and ***OpenRegistration.statusDetailText*** are to be filled-in by the RandoNode showing which fields are mismatching with the original patient data.

ii. If the ***OpenRegistration.patientId*** field is not filled-in then, match the submitted demography data with that of all previous patients in the group's database.

iii. if one or more patients match with the submitted demography data then,

1. Create a new ***ExistingPatient*** object (or objects).
2. Fill in the attributes of ***ExistingPatient*** object(s).
3. Add this object to ***RegistrationResponse.existingPatientList*** field after creating a new array of ***existingPatientList***.
4. Return the function after setting ***OpenRegistration.status*** value as any one of the following values depending on the matching scheme used by the group as well as if the match was found in this study or in another study.

```
{"PT_IS_DUPLICATE",
"PT_IN_OTHER_STUDY",
"PT_POSSIBLY_DUPLICATE",
"PT_POSSIBLY_IN_OTHER_STUDY"}
```

iv. If no previous patient is matched with the demography data then.

1. Set the attribute of ***RegistrationResponse.existingPatientList*** as null.
2. Proceed with any other validation of the demography data.
3. If demography validation is successful, return the function with ***OpenRegistration.status*** = "SUCCESS" or else return the function with ***OpenRegistration.status*** = "FAILURE".

b) Else If ***OpenRegistration.userResponse*** == "PT_SAME_AS_EXISTING_PT" || "PT_CONFIRMED_NEW" || "NOT_APPLICABLE" then proceed with any other validation of the demography data as described in item 3.a.iv.2.

4. If (***openRequest.operation*** == "VALIDATE_ALL_DATA") then

a) Assume this as an EC form submission (for example, this can be the existing ***doValidate()*** call from ***doRegister()***) and perform all the validations starting with the validation mentioned in step 3.a above.

b) If a previous patient is matched with the demography data then return the function as described in step 3.a.i.4 above.

c) Otherwise continue to validate demography data followed by EC data. If all validations are successful then return the function without any error indication with ***OpenRegistration.status*** = "SUCCESS" flag. If the validation of demography or EC data fails then return the function with ***OpenRegistration.status*** = "FAILURE" flag.

Shown below is a simplified workflow diagram showing the above sequence of operations.

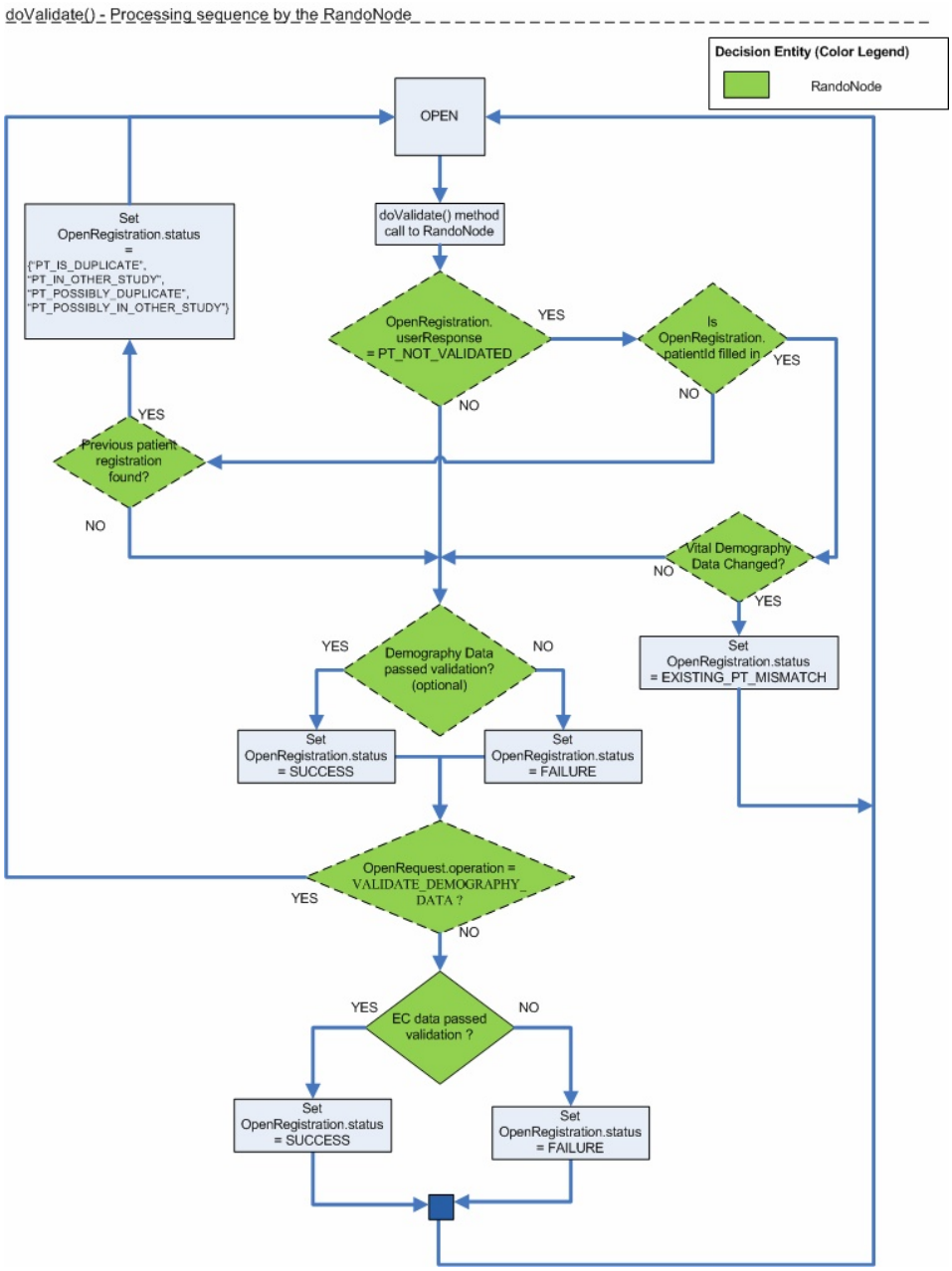

Figure 12: Workflow diagram for doValidate() function processing by the RandoNode

Setting the ***RegistrationResponse.existingPatientList*** attribute is the key for this verification. If this attribute is ***null*** then OPEN will assume that there are no existing patients matching the same demography data

and will proceed to the next step in the registration process.

If ***RegistrationResponse.existingPatientList*** attribute is not null, then OPEN will retrieve the information from that object and will display a meaningful message to the site registrar using the ***OpenRegistration.status*** attribute value supplied by the RandoNode.

5.4.2. Changes to doRegister() web method

When the RandoNode receives the doRegister() method call, they must call the doValidate() method to do the existing patient verification.

Part III – RandoNode Implementation Details (JAVA RandoNode)

6. Summary of Changes

OPEN portal allows registrars to select an administrative person for multiple roles. RandoNode 1.0 used to support a single role for each person. RandoNode 1.1 is modifying the existing logic to accommodate multiple roles similar to OPEN portal. RandoNode 1.1 also provides few utility methods.

7. Details Of Changes

7.1. Structure Change To User Class

To accommodate the requirement mentioned above, some structural changes are made to User.java class. This class had a member variable openUserType to indicate user role. RandoNode 1.1 is replacing the variable openUserType with openUserTypeList as shown below. openUserTypeList is a list of String.

```
public class User{
    public long id;
    public String oid = "";
    public String ctepId = "";
    public Location location;
    public String locationOid;
    public String firstName = "";
    public String lastName = "";
    public String email = "";
    public String phone = "";
    public String orgnaization = "";
    public String loginName = "";
    public String displayName = "";
    public String userType = "";
    /**
     * openUserTypeList replaces the variable openUserType, since user
     * can have multiple roles. This variable returns all the roles in
     * form of a list.
     * // public String openUserType = "";
     */
    private ArrayList<String> openUserTypeList = new ArrayList<String>();
```

Group developers need to modify their code to refer to openUserTypeList instead of openUserType. Accessor functions are provided in User.java class as shown below to retrieve openUserTypeList.

```
public ArrayList<String> getOpenUserTypeList() {
    return openUserTypeList;
}

public void setOpenUserTypeList(ArrayList<String> openUserTypeList) {
    this.openUserTypeList = openUserTypeList;
}
```

7.2. New Utility Methods

Few new utility methods were introduced in ClinicalDataUtil.java and MetaDataUtil.java. Please see the detail below.

Method	Class	Description
getUsersByOpenUserType	ClinicalDataUtil.java	**Input:** String ; **Output:** ArrayList<User> This public method accepts openUserType as input and returns the List of users associated with the openUserType provided.
getAvailableOpenUserTypes	ClinicalDataUtil.java	**Input:** None ; **Output:** ArrayList<String> This public method returns the List of all openUserType available within current registration request.
isCTSUDataEntryPerson	ClinicalDataUtil.java	**Input:** None; **Output:** boolean This public method determines whether the data entry person is a CTSU person and returns true or false based on that.
getStandardDemographyObject	ClinicalDataUtil.java	**Input:** None; **Output:** Demography This public method returns standard demography data if present in request. This method does not retrieve specific demography data, only used to retrieve standard demography.
getCodeListItemByCodedValue	MetaDataUtil.java	**Input:** String, String ; **Output:** CodeListItem This public method returns the CodeListItem based on the CodeList oid and coded value provided.

7.3. How to Install/Upgrade To RandoNode 1.1.0.0 Build

Please refer to the document readme.doc of RandoNode 1.1.0.0 for installation/up gradation guide. Readme.doc document can be located within the RandoNode.Zip distributed to groups.

Part IV – RandoNode Implementation Details (.Net RandoNode)

The following are the major changes with this version

1. A new web method named ***getPatientData()*** added to the RandoNode so that OPEN can use it to get the patient data for a given patientId.
2. Introduced new web ***methodgetVersion()*** so that OPEN can obtain the version of the RandoNode. This method is not intended to be overridden.
3. Modified ***doValidate()*** and ***doRegister()*** methods to add validation for the demography data – to verify if this patient already exist in this protocol (duplicate patient) or any other protocol (existing patient).
4. Implemented ***doRegisterTest()*** and ***doCredential()*** methods.
5. Introduced verification of operation attribute in all the web methods.
6. Introduced Demography and ***ExistingPatient*** objects within ***RegistrationResponse*** object.
7. Introduced new attributes within ***OpenRegistration*** object and demonstrated their application in web methods.
8. Modified the valid values of ***OpeRegistration.status*** attribute.
9. Logging functionality refactored for more flexibility when extending the RandoNodes.
10. More code refactoring and cleanup.
11. Several utility functions are added for easy extension of the RandoNode.

Introduced following utility methods in *ClinicalDataUtil* class.

1. ***getStandardDemographyObject()*–** Retrieves Standard Demography Data if present in the request.

Part V – Resources

8. Appendix I

Table 7: Allowed valid values for various Demography attributes

#	Attribute	CDE Public ID	Valid Values	Comment
1.	ethnicity	2192217	Hispanic or Latino Not Hispanic or Latino Not reported Unknown	
2.	gender	2200604	Female Gender Male Gender Unknown Unspecified	
3.	countryOfResidence	315	The valid values specified in CDE Browser for the CDE ID 2006183 will be used. Please access reference **Error! Reference source not found.** for additional information.	
4.	race	2192199	American Indian or Alaska Native Asian Black or African American Native Hawaiian or other Pacific Islander Not Reported Unknown White	Can have multiple races for the same person
5.	methodOfPayment	2003309	PRIVATE INSURANCE MEDICARE MEDICARE AND PRIVATE INSURANCE MEDICAID MEDICAID AND MEDICARE MILITARY OR VETERANS SPONSORED NOS MILITARY SPONSORED (INCLUDING CHAMPUS &TRICARE) VETERANS SPONSORED SELF PAY (NO INSURANCE) NO MEANS OF PAYMENT (NO INSURANCE) OTHER Unknown	Can have multiple payment options for the same person
6.	educationLevel	2674076	10th Grade 11th Grade 12th Grade No Diploma 1st Grade 2nd Grade	This is an optional field

#	Attribute	CDE Public ID	Valid Values	Comment
			3rd Grade 4th Grade 5th Grade 6th Grade 7th Grade 8th Grade 9th Grade Academic Doctorate Degree Associate Degree Bachelor Degree Don't Know General Equivalency Diploma High School Graduate Kindergarten Master's Degree No Formal Schooling Preschool Professional Doctorate Degree Refused Some College, No Degree VoTech Program	
7.	educationalAttainment	2681552	Bachelor's Degree Doctoral degree or professional degree Graduate or professional degree High school graduate (including equivalency) Master's Degree Not high school graduate Some college or associate degree	This is an optional field
8.	maritalStatus	2188083	Divorced Domestic Partnership Married Never Married Separated Widowed	This is an optional field