IT-9013

Comprehensive Data Resource (CDR) Architecture & Design

Version 1.01

11/7/2016

Version History

This document combines information from a variety of existing sources in describing the design and context under which the Comprehensive Data Resource operates. This document is consistent with version 6.0 of the CDR.

Version	Author	Effective Date	Description of Change
1.00	William Lander	11/4/2016	Initial Version
1.01	William Lander	11/7/2016	Administrative Update

TABLE OF CONTENTS

Table of Figures

Table of Tables

1 Introduction

Leidos Biomedical Research, under the direction of the NCI BBRB, developed the Comprehensive Data Resource (CDR). CDR addresses the challenges of collection process of data regarding tissues gathered in the early stages of the biospecimen lifecycle.[1] These data include information about potential candidates; their eligibility criteria and consent; medical/surgical procedures used; and acquisition processing, handling, and storage. As the focus for biospecimen-based studies of cancer has turned to the molecular level, applying stringent laboratory techniques in the lifecycle is more important than ever.

The initial design of the CDR reflects the changing requirements of real-world data collection.

- As the collection process progresses, the need for different or more detailed information may arise. Changing needs necessitate rapid changes (on the order of days) in forms, Web pages, workflows, and database structure.
- Support various review cycles of detail recording during the collection process. This includes reviews for data entry or processing errors (data management) and specimen quality (both internal and external pathology review).
- Support geographic separation between those working on the supported projects during data collection, while protecting personally identifiable information (PII) and protected health information (PHI). The collection and processing sites may change during the course of the project.
- Mnimize the learning curve for users. The interface must reflect the historical paper-based forms process.
- Supply definitions of terms interactively, using a standard, approved vocabulary. Specialized terms often have different meanings to different institutes. Terms must be used consistently, and their definitions must be available to the end user. This allows researchers to properly analyze research results.

The first project producing requirements for the CDR was the National Institutes of Health (NIH) Common Fund's Genotype-Tissue Expression (GTEx) program.[2] GTEx collected 944 cases with more than 6,700 individual specimens.

[1] http://biospecimens.cancer.gov/researchnetwork/lifecycle.asp

[2] https://commonfund.nih.gov/GTEx/index

The CDR code was later extended to cover a second project, Biospecimen Pre-analytical Variables (BPV) Program.[3] BPV collected information on more than 300 cases, with specimens covering four different cancer types (and associated normal tissue) from four different tissue source sites.

The current architecture described in this document was designed to support these two experiments, but it can be extended to many others.

1.1 Purpose of This Document

This document gives an overview of the architecture and system design which went into the CDR development. This gives the development team guidance on the resource's architecture in case the need arises for further changes, updates, or extensions. Its intended audience is the project manager, project team, and development team. Some portions of this document, such as the user interface (UI) (see section 5.5), may on occasion be shared with users and with other stakeholders whose input on or approval of the UI is needed.

[3] http://biospecimens.cancer.gov/programs/bpv/default.asp

2 High-Level View of the CDR

2.1 CDR Capabilities

The CDR is a distributed multi-tenant bioinformatics platform, custom-built to support specimen collection, clinical data entry, coordination of specimen logistics, and curation and aggregation of study data. Its capabilities, which reflect the needs of the supported projects, include the following:

- Allow remote users (e.g., clinicians, researchers, support staff) to securely enter, revise, and review data about biospecimen collection through a standard (HTTPS) Web interface using a series of electronic forms with a sophisticated role-driven workflow.
- Connect to remote systems via Web service application program interfaces (APIs), such as laboratory information management systems (LIMS), whole-slide imaging systems, and molecular analysis systems. The APIs provide for authenticated (RESTful interfaces) real-time, two-way data, transferring content across a variety of data types. The CDR has no capability for using a database-to-database direct connection to other database servers.
- Trigger responses to automatically communicate timely information to project managers and data analysts.
- Assist with quality assurance by auditing process flows through data management and pathology teams.
- Control display of PII based on user entitlements and roles.
- Analyze and aggregate data and generating reports in real time.

Figure 1, taken from GTEx source materials, shows the CDR acting as the project's data hub, storing information about the collections from the biospecimen source sites (BSSs); transferring and processing information through the Cancer Human Biobank (caHUB) Comprehensive Biospecimen Resource (CBR); and handing off information on biospecimens to the brain bank and Laboratory, Data Analysis, and Coordinating Center (LDACC). Detailed descriptions of the components appear in Appendix D. The flow of tissue is shown in blue, and the flow of information about those tissues is shown in gray. When the BSSs collect tissues, they use the forms hosted by the CDR to describe the case, biospecimens, and other clinical information. As the biospecimens transfer to the CBR, the CDR receives inventory information on them (through Web service messages) from the informatics system at the CBR. This instantiates workflows and creates triggers for data managers and pathologists to review cases. Once the tissues are processed, the CBR records subsequent transfer of materials to other specialized labs, including the LDACC for molecular studies. Web services and users enter results of various tests at these specialized labs into the CDR, providing comprehensive study information.

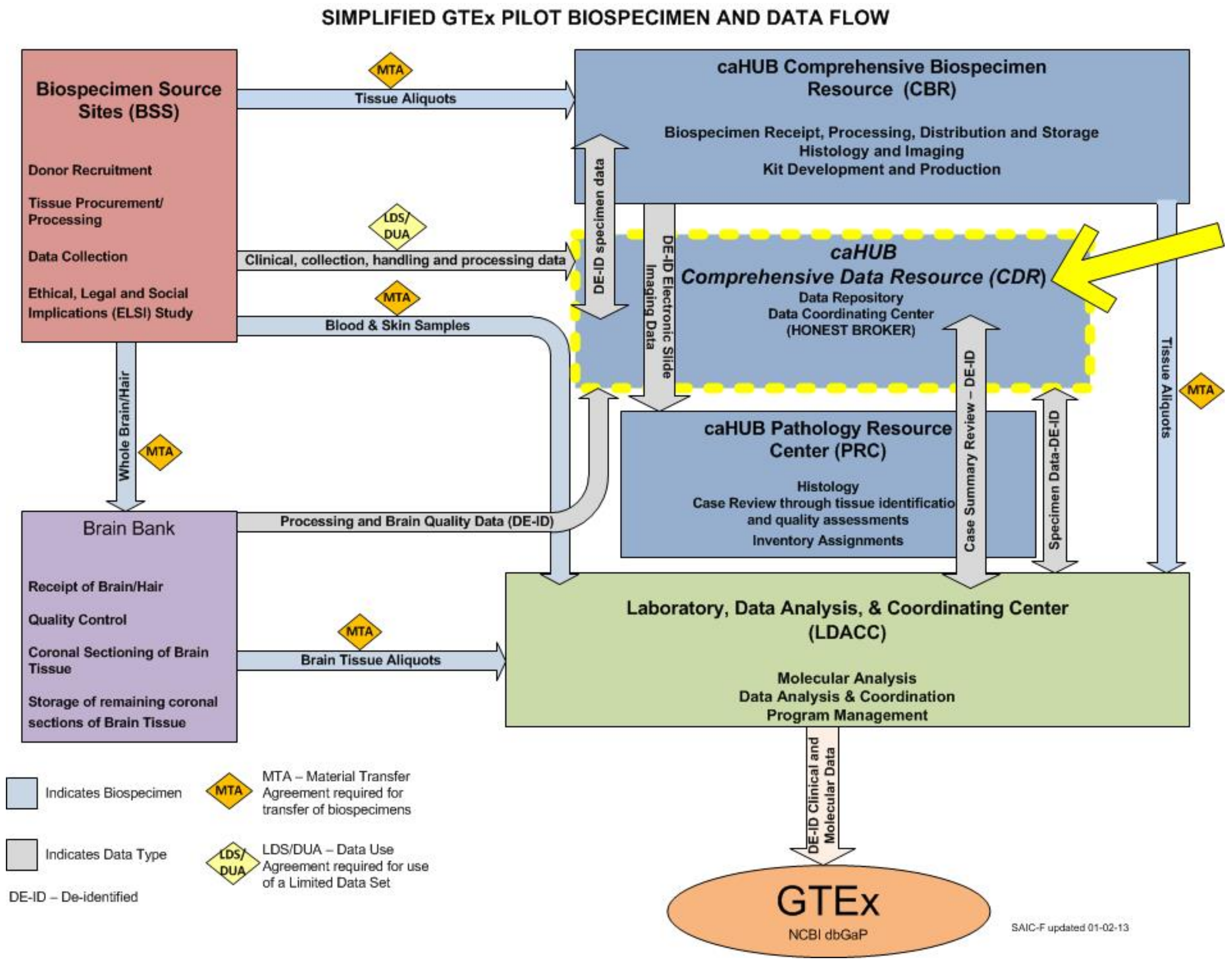
SIMPLIFIED GTEx PILOT BIOSPECIMEN AND DATA FLOW
Biospecimen Source Sites (BSS)
Donor Recruitment
Tissue Procurement/ Processing
Data Collection
Ethical, Legal and Social Implications (ELSI) Study
MTA
Tissue Aliquots
LDS/ DUA
Clinical, collection, handling and processing data
Blood & Skin Samples
caHUB Comprehensive Biospecimen Resource (CBR)
Biospecimen Receipt, Processing, Distribution and Storage
Histology and Imaging
Kit Development and Production
DE-ID specimen data
DE-ID Electronic Slide Imaging Data
caHUB Comprehensive Data Resource (CDR)
Data Repository
Data Coordinating Center
(HONEST BROKER)
Tissue Aliquots
Whole Brain/Hair
caHUB Pathology Resource Center (PRC)
Histology
Case Review through tissue identificatio and quality assessments
Inventory Assignments
Case Summary Review – DE-ID
Specimen Data-DE-ID
Brain Bank
Receipt of Brain/Hair
Quality Control
Coronal Sectioning of Brain Tissue
Storage of remaining coronal sections of Brain Tissue
Processing and Brain Quality Data (DE-ID)
Brain Tissue Aliquots
Laboratory, Data Analysis, & Coordinating Center (LDACC)
Molecular Analysis
Data Analysis & Coordination
Program Management
DE-ID Clinical and Molecular Data
Indicates Biospecimen
Indicates Data Type
DE-ID – De-identified
MTA – Material Transfer Agreement required for transfer of biospecimens
LDS/DUA – Data Use Agreement required for use of a Limited Data Set
GTEx
NCBI dbGaP
SAIC-F updated 01-02-13

Figure 1: The CDR in the Context of the GTEx Study

A more exact description of how the CDR supports the data workflows in GTEx appears in the SOP GX3-OP-9001, found in the study documents.

Figure 2 illustrates the data flow in the BPV project. Many of the data exchanges in BPV are similar to those in GTEx, but many new forms were created and the CDR was extended, supporting linkage to Aperio image viewers.

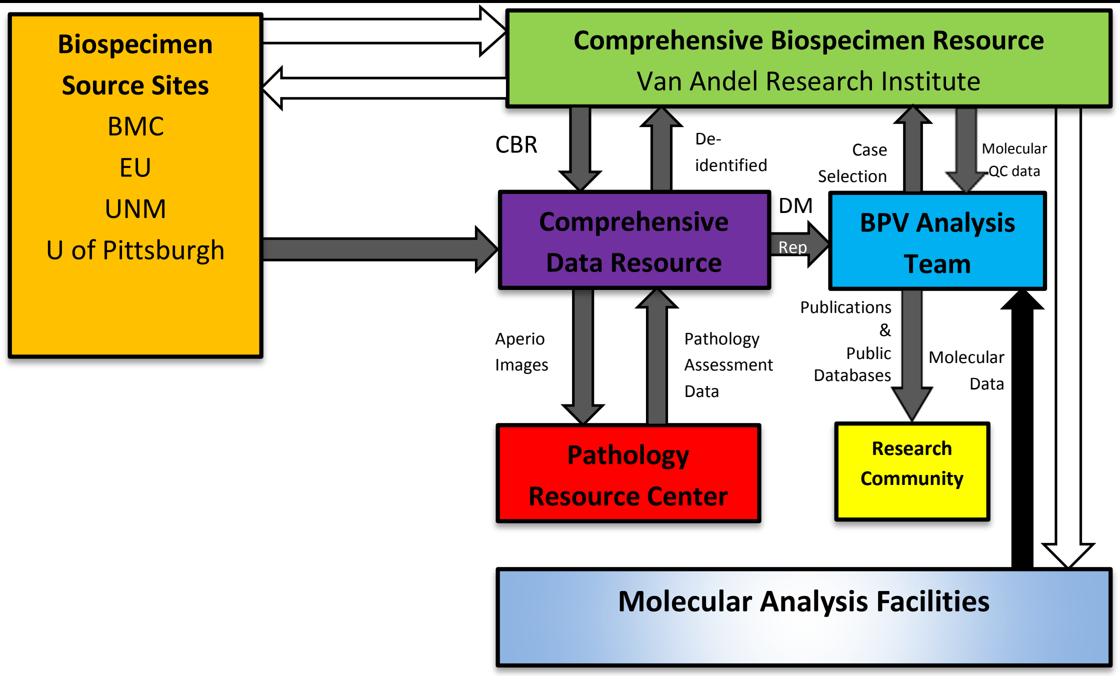
Biospecimen Source Sites
BMC
EU
UNM
U of Pittsburgh
Comprehensive Biospecimen Resource
Van Andel Research Institute
CBR
De-identified
Case Selection
Molecular QC data
Comprehensive Data Resource
DM
Rep
BPV Analysis Team
Aperio Images
Pathology Assessment Data
Publications & Public Databases
Molecular Data
Pathology Resource Center
Research Community
Molecular Analysis Facilities

Figure 2: Data Flow in the BPV Project

Projects as complex as those the CDR supports require a continuous management effort. The CDR supports project management by providing a variety of customizable reports to help project managers monitor the effectiveness and quality of the overall biospecimen and data collection process. To help monitor and control the quality of the products, the CDR supports internal reviews by both the data management team (for data accuracy and completeness) and the PRC (for whole-slide image reviews of the biospecimens). The results of these internal reviews are recorded in the CDR as part of the case study permanent record.

2.2 Using the CDR

The CDR is hosted at the Frederick National Laboratory for Cancer Research (FNLCR). Providing user access within an existing study is subject to NIH and National Cancer Institute (NCI) electronic use policies as well as program training requirements, such as Health Insurance Portability and Accountability Act (HIPAA) and system training. Users are provided accounts based on predefined study roles.

Although setting up and tailoring a study requires custom coding and configuration, the CDR's core data model, security layer, and API interfaces are robust and easily extended to new studies. The level of effort involved in making changes to the CDR will vary with a new project's requirements. Grails was chosen as the implementation language because it reduces the time needed for development by supporting automated generation of Web interfaces and automated management of the database schema, persistence, and searching. No database administrators are needed once the supporting Oracle instance is up and available; changes in the schema are managed internally by Grails Object Relational Mapping (GORM). All database access is based on Hibernate 3, which manages table definition updates as well as content updates.

Data exchanges between the CDR and external systems, such as a LIMS, or molecular analysis facilities require agreement on an XML interface. The CDR operates RESTful Web service interfaces with XML payloads tailorable to different payload structures based on scientific and program requirements. Details of the XML interface are described in the Interface Control Document (ICD – IT-9004 – CDR Data Services Work Instructions and User Guide, found in the ancillary documents folder).

3 General Overview and Design Guidelines/Approach

This section describes the principles and strategies that were used as guidelines when designing and implementing the CDR.

3.1 Assumptions, Constraints, and Standards

The CDR architecture is designed for maximum flexibility to meet changing requirements. As such, it uses standard services (e.g., Oracle database services, RESTful Web service interfaces, XML, and JSON data exchanges) robustly to create a framework into which a variety of documents (each with different data, vocabulary, and constraints) may be "plugged." The CDR UI is currently undergoing Section 508 standards testing to ensure that it meets Department of Health and Human Services requirements.[4]

The architecture reflects the need to store PII and PHI in an environment where not all users have a need to know all information. For this reason, it includes a variety of roles that can be assigned to users, limiting their ability to access sensitive information. Section 4.4 discusses this aspect of the architecture in detail.

3.2 Alignment with Federal Enterprise Architecture (FEA)

The CDR is aligned with the FEA at the application scope level. The application scope level focuses on the one application (in this case, the CDR), which uses several systems and services from various institutional and governmental organizations, including websites, databases, email, RESTful Web services, and other support applications.

[4] http://www.hhs.gov/web/section-508/compliance-and-remediation/framework/index.html

4 Architecture Designs

The LBR Biospecimen Research Group built the CDR as an enterprise-level application built around the motto “Science First.” The technologies chosen for building the CDR, including the open-source Grails framework and other proven open-source technologies, facilitated a faster development time through rapid and agile software methods. The architecture leverages existing FNLCR database assets and infrastructure, reducing costs and overhead. Section 4.2 describes this work.

As the project requirements evolved, the CDR’s data model, features, and interfaces were upgraded and enhanced. Iterative software releases were coordinated with scientific project milestones and program phases.

4.1 Logical View

The CDR consists of two modules: CDR-DS (for “data services”) and CDR-AR (for “analysis and reporting”). CDR-DS provides secure user access to case and biospecimen sample data based on pre-defined roles and privileges. It uses dynamic content redaction to restrict PII and PHI to a limited data set (LDS) with access given only to authorized users. The overall logical flow of information is shown in Figure 1. Intuitive graphic UIs for the BSSs streamline data entry workflow by strictly following standard operating procedures (SOPs) for sample collection and processing. Contextual automated data checks and business rule validations confirm data integrity and adherence to biospecimen collection and preservation SOPs. Web services APIs allow the PRC to access digital imaging data from whole-slide images housed remotely at the CBR. APIs connect to the CBR’s LIMS for real-time sample inventory data. De-identified GTEx data are provided through a private API with the Broad Institute (LDACC) before their final release into the database of Genotypes and Phenotypes (dbGaP). The CDR analytics and reporting module, the CDR-AR, supports data analysis and aggregation, report generation, and real-time operational data snapshots.

Users of the Web forms interface from collaborating institutions are granted access through application-specific user accounts. They can then enter data and access existing data entered under their institutions’ activities. Administrative, Data Manager, Project Manager, and read-only roles are assigned as needed through the account provisioning process. Each RESTful Web interface client (e.g., institute sharing data over the RESTful interface) also receives a unique username and password.

User credentials authorization comes through either NIH LDAP accounts or study-specific Spring-approved access. Data Managers do account provisioning. Role access comes from a person-in-the-loop evaluation of an individual’s duties and responsibilities. Spring Security, built into Grails, enforces role limits. Figure 3 shows the Spring configuration file and the safeguards and privileges. The criteria determining which user accounts should have access to each role are study specific and are described in the appropriate study’s SOPs. In no case does the CDR give users direct access to files, networks, or other features of the host network.

Any URL not in this list is denied

```
/appSetting/** etc., is the URL
['ROLE_ADMIN','ROLE_SUPER'], is the Role
grails.plugin.springsecurity.controllerAnnotations.staticRules = [
  //system setting controllers
  '/user/**': ['ROLE_ADMIN','ROLE_SUPER'],
  '/role/**': ['ROLE_ADMIN','ROLE_SUPER'],
  '/userRole/**': ['ROLE_ADMIN','ROLE_SUPER'],
  '/securityInfo/**': ['ROLE_ADMIN','ROLE_SUPER'],
  '/controllers.gsp':['ROLE_ADMIN','ROLE_SUPER'],
  '/backoffice/**':['ROLE_ADMIN','ROLE_SUPER'],
  '/auditLogEvent/**':['ROLE_ADMIN','ROLE_SUPER'],
  '/userLogin/**': ['ROLE_ADMIN','ROLE_SUPER','ROLE_DM'],
  '/privilege/**':['ROLE_ADMIN','ROLE_SUPER','ROLE_DM','ROLE_PRC','ROLE_LDS'],
  '/tissueType/**':['ROLE_ADMIN','ROLE_SUPER','ROLE_DM'],

  //leave these alone. these rules are needed for everything to work properly
  '/login/**': ['IS_AUTHENTICATED_ANONYMOUSLY'],
  '/logout/**': ['IS_AUTHENTICATED_FULLY'],
  '/register/**': ['IS_AUTHENTICATED_ANONYMOUSLY'],
  '/plugins/**': ['IS_AUTHENTICATED_ANONYMOUSLY', 'IS_AUTHENTICATED_FULLY'],
  '/images/**': ['IS_AUTHENTICATED_ANONYMOUSLY'],
  '/css/**': ['IS_AUTHENTICATED_ANONYMOUSLY'],
  '/js/**': ['IS_AUTHENTICATED_ANONYMOUSLY'],

  //webapp controllers
  '/home/**': ['IS_AUTHENTICATED_FULLY'],
  '/appSetting/**': ['IS_AUTHENTICATED_FULLY'],
  '/caseRecord/**': ['IS_AUTHENTICATED_FULLY'],
  '/candidateRecord/**': ['IS_AUTHENTICATED_FULLY'],
  '/specimenRecord/**': ['IS_AUTHENTICATED_FULLY'],
  '/slideRecord/**': ['IS_AUTHENTICATED_FULLY'],
  '/study/**': ['IS_AUTHENTICATED_FULLY'],
  '/organization/**': ['ROLE_DM', 'ROLE_ADMIN'],
  '/bss/**': ['IS_AUTHENTICATED_FULLY'],
  '/user/**': ['IS_AUTHENTICATED_FULLY'],
  '/role/**': ['IS_AUTHENTICATED_FULLY'],
  '/activityType/**':['IS_AUTHENTICATED_FULLY'],
  '/activityEvent/**': ['IS_AUTHENTICATED_FULLY'],
  '/activitycenter/**': ['IS_AUTHENTICATED_FULLY'],
  '/textSearch/**': ['IS_AUTHENTICATED_FULLY'],
  '/textSearch/index_all': ['ROLE_ADMIN'],
  '/query/**':['ROLE_BSS_UUU','ROLE_BSS_CCC',
'ROLE_DM','ROLE_SUPER','ROLE_ADMIN','ROLE_ORG_VARI','ROLE_ORG_BROAD','ROLE_ORG_MBB'],
  '/fileUpload/**': ['ROLE_ADMIN','ROLE_BSS', 'ROLE_DCC'],
  '/caseAttachmentType/**':['ROLE_ADMIN'],
  '/prcReport/**': ['ROLE_PRC','ROLE_ADMIN'],
  '/prcReport/view': ['ROLE_PRC','ROLE_ADMIN', 'ROLE_DCC'],
  '/healthHistory/**': ['ROLE_BSS', 'ROLE_DCC'],
  '/socialHistory/**': ['ROLE_BSS', 'ROLE_DCC'],
  '/surgeryAnesthesia/**': ['ROLE_BSS', 'ROLE_DCC','ROLE_ADMIN'],
  '/tissueGrossEvaluation/**': ['ROLE_BSS', 'ROLE_DCC','ROLE_ADMIN'],
   '/generalMedicalHistory/**': ['ROLE_BSS', 'ROLE_DCC'],
   '/cancerHistory/**': ['ROLE_BSS', 'ROLE_DCC'],
   '/medicationHistory/**': ['ROLE_BSS', 'ROLE_DCC'],
   '/screeningEnrollment/**': ['ROLE_DCC','ROLE_BSS'],
   '/consentVerification/**': ['ROLE_DCC','ROLE_BSS'],
   '/demographics/**': ['ROLE_DCC','ROLE_BSS'],
   '/blood/**': ['ROLE_DCC','ROLE_BSS'],
   '/tissueReceiptDissection/**': ['ROLE_DCC','ROLE_BSS'],
   '/rest/**': ['ROLE_ADMIN']
]
```

Figure 3: Spring Security Configuration File, Showing Roles and Restrictions

The CDR connects various tissue procurement sites (BSSs), remote LIMS (CBR), and molecular analysis (LDACC) APIs, as shown in Figure 4. Hundreds of GTEx cases are at various stages in the data flow pipeline shown in Figure 2. Pathologists and clinical data managers curate and review the data entered by staff at collection sites (BSSs) in real time. Biospecimen logistical data and metadata are transferred from the CBR to the CDR in real time. Data managers control the release of data to the LDACC, where it is associated with the genotype and expression data and then exported to dbGaP, becoming available to the research community.

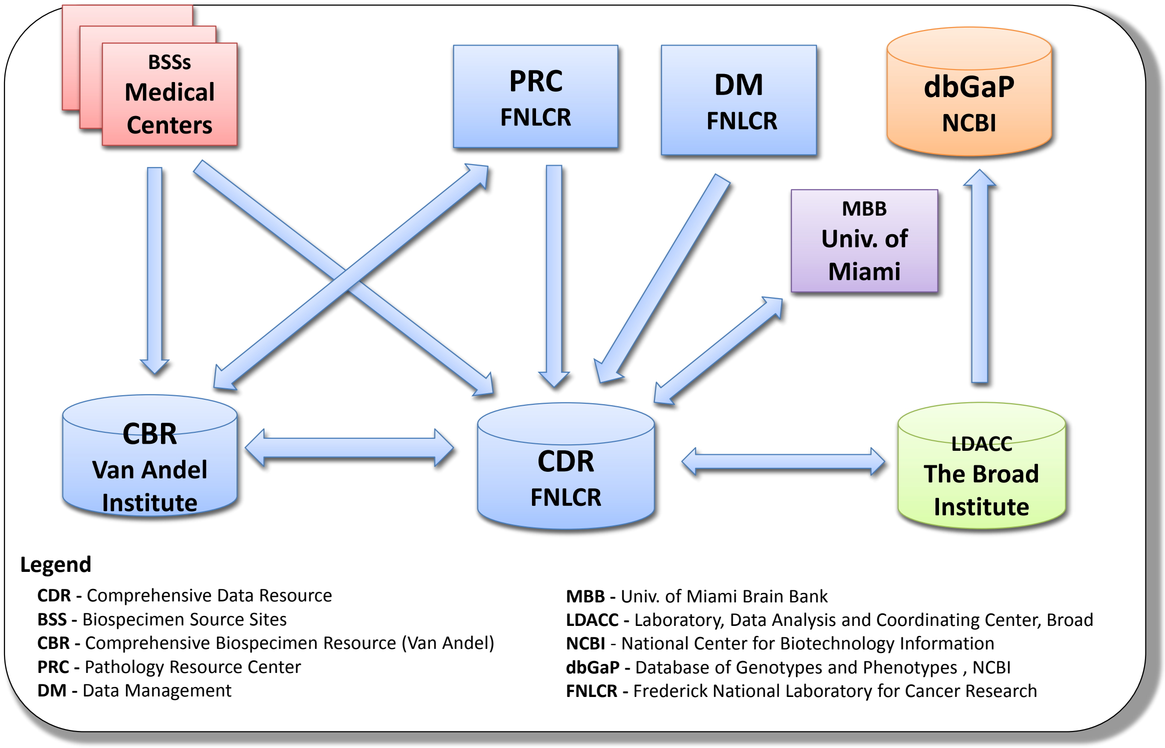
BSSs
Medical Centers
PRC
FNLCR
DM
FNLCR
dbGaP
NCBI
MBB
Univ. of Miami
CBR
Van Andel Institute
CDR
FNLCR
LDACC
The Broad Institute
Legend
CDR - Comprehensive Data Resource
BSS - Biospecimen Source Sites
CBR - Comprehensive Biospecimen Resource (Van Andel)
PRC - Pathology Resource Center
DM - Data Management
MBB - Univ. of Miami Brain Bank
LDACC - Laboratory, Data Analysis and Coordinating Center, Broad
NCBI - National Center for Biotechnology Information
dbGaP - Database of Genotypes and Phenotypes , NCBI
FNLCR - Frederick National Laboratory for Cancer Research

Figure 4: CDR External Data Flows

The Van Andel Institute uses a program called BRIMS to automate processes at the CBR. BRIMS and the CDR exchange information via a RESTful interchange. Representational state transfer (REST) uses the underlying standards of the World Wide Web Consortium that enable the transfer of information, ignoring the details of component implementation and protocol syntax to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements. Details of the information exchange are documented in IT-9004 – CDR Data Services Work Instructions and User Guide, found in the ancillary documents folder. There is no direct database access between institutes or between the CDR and other institutes.

4.2 Hardware Architecture

The CDR is hosted on a number of Virtual Machines (VMs) located at the FNLCR Advanced Biomedical Computer Center. As shown in Figure 5, separate VMs are used to host both modules of the CDR (CDR-DS and CDR-AR) and the Apache front-end. The CDR-DS and the CDR-AR share a common instance of Oracle. They communicate only through changes in the Oracle tables. Because the AR version has no OLAP (online analytical processing)–type functionality, database demands are low; sharing Oracle instances has substantial impact to the users, and reports must be up to date.

The Simple Mail Transfer Protocol (SMTP) mail server is a shared mail server for all of NIH; it is not dedicated to CDR activities.

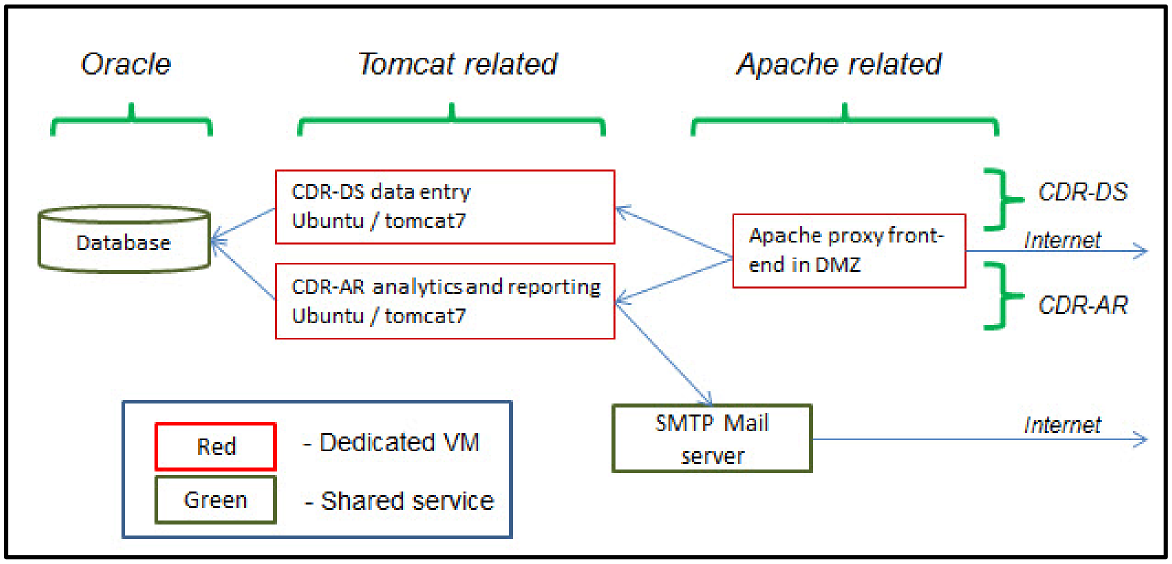

Figure 5: Virtual Machine Network Architecture

To convey the computational power necessary for the CDR working with the current user load, Table 1 shows the current VM configurations for the dedicated resources. As the scope of CDR activities grows, the VMs can be reconfigured, adding resources (e.g., memory, processors) without the need to update software.

Table 1: Current Virtual Machine Configurations

VMWare ESX Host	RAM	CPU	Disk storage capacity
CDR-DS	4,096 MB	2—Westmere Xeon Core i7	68 GB
CDR-AR	4,096 MB	2—Westmere Xeon Core i7	72 GB
Apache	2,048 MB	1—Westmere Xeon Core i7	15 GB
Oracle Instance	4,096 MB	2—Westmere Xeon Core i7	unlimited

4.3 Software Architecture

Grails provides the software architecture as a framework, shown in Figure 6 and Figure 15. The definition of Grails is maintained at http://grails.github.io/grails-doc/latest/guide/spring.html. Like other Java languages, Groovy, the compiled Grails language, runs in a Java VM. For Web applications, this Java VM is shared with the Tomcat application container. Figure 6 shows how Spring supports elements of the enterprise service layer; SiteMesh is a decorator engine, supporting view layouts for generating Web page displays.

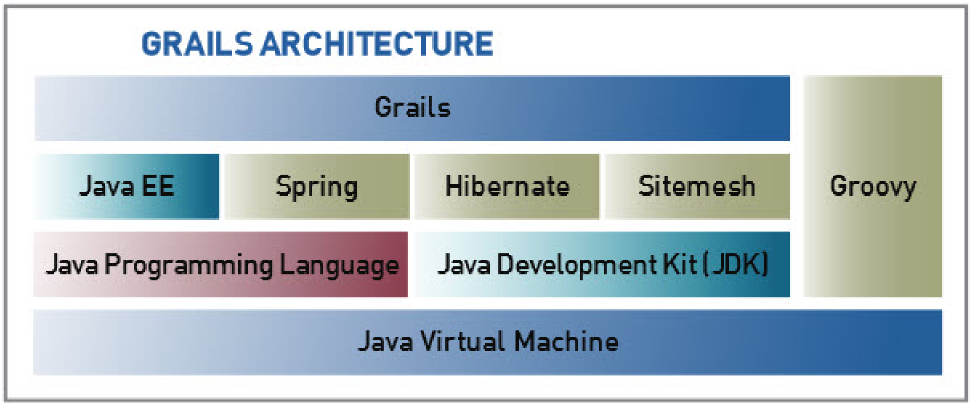

Figure 6: Grails Framework Architecture

4.3.1 CDR Domain Class Model

The heart of the CDR is its domain model, the set of domain classes and their relationships. Domain classes hold the persistent objects in the database. They are linked together through relationships: one-to-one, one-to-many, or many-to-many. In Figure 6, these domain classes reside in the Grails level.

A domain class represents persistent data and, by default, is used to create a table in a database. The name of the domain class (the “model” in model-view-controller [MVC]) is the same as the name of the corresponding controller and views. The location of the file in the CDR directory structure, along with the domain class name, gives the intent of the code in a file. One of the development benefits of Grails is that, by default, all the capabilities of a basic CRUD (create, read, update, and delete) application are already available. When a new model (such as a new form) is needed, a domain class must be added to the source files. The NetBeans IDE supports automatic code development when domain classes are added. After defining a domain class in NetBeans, right click on it, and choose “generate all.” Grails will generate the controller and default views for the new domain class.

The term “data warehouse” has a particular meaning within the CDR. Commonly, a data warehouse gathers data accumulated from a wide range of sources within a company and used to guide management decisions. The CDR data warehouse does not collect data from sources outside the CDR, nor does it directly support management decisions. The reporting done by the CDR uses real-time data from a single instance of the database. The CDR uses periodic processes (e.g., overnight jobs) in limiting the complexity of database queries, and reducing the response time needed for some reports.

Table 2 enumerates and describes the domain classes making up the CDR.

Table 2: CDR Domain Classes

Domain Class	Description
`accountAction`	Implements the various actions needed to update the capabilities, privileges, and roles of CDR users
`accountStatus`	Implements the reporting of accounts and their privileges and roles
`applicationListing`	Supports the creation and display of lists of applications
`appRequest`	Supports the creation, display, editing, and listing of users' accounts, statuses, and associated applications
`appUsers`	Supports user creation, activation/deactivation, and roles
`bpvBloodForm`	Supports the use of the BPV study's blood collection form, including various tables, displays, and user inputs to field values
`bpvCaseQualityReview`	Supports the creation, editing, listing, and display of various factors used in the case quality review process
`bpvClinicalDataEntry`	Supports the creation, updating, listing, and display of information related to the Clinical Data form
`bpvConsentEnrollment`	Supports the creation, updating, listing, and display of information related to the Consent for Enrollment form. Consent is necessary before collections and related activities can proceed.
`bpvElsiCrf`	Supports the BPV Ethical, Legal, and Social Implications (ELSI) form, used in research on why people do or do not participate in studies
`bpvLocalPathReview`	Supports the pathology review done by the tissue source sites in BPV
`bpvPrcPathReview`	Supports the independent pathology review done by the PRC in BPV
`bpvScreeningEnrollment`	Supports entry, updating, and display of screening information for enrolling new (potential) participants in BPV. This screening information is a critical part of determining the acceptability of a specific case.
`bpvSixMonthFollowUp`	Supports the collection, display, and updating of information gathered in the six-month post-collection time frame
`bpvSlidePrep`	Supports the display and updating of information about the various processes used in turning collections into meaningful slides, including staining and related activities
`bpvSurgeryAnesthesiaForm`	Supports the display and recording of information about the types and quantities of anesthesia used on a patient during surgery
`bpvTissueForm`	Supports the display and recording of information about the number and types of tissues collected from a particular case in BPV

Domain Class	Description
bpvTissueGrossEvaluation	Supports the display and recording of information entered by the local Biospecimen Source Site Pathologist, describing the macroscopic evaluation of the tissue
bpvTissueProcessEmbed	Supports processing and embedding of tissue in formalin-fixed, paraffin-embedded blocks
bpvTissueReceiptDissection	Supports the display and recording of information on the handling of tissue aliquots received from the operating room
bpvWorkSheet	Supports data entry and barcode scanning with automatic timestamps as specimens are entered in the protocols for delay to fixation and time in fixative
brainBankFeedback	Supports the display and recording of information by the University of Miami Brain Endowment Bank brains supplied by the GTEx program for its studies
cahubAcceptable	Supports the creation, display, and editing of the acceptability of any tissue. Acceptability is a function of many dimensions, including acceptance in study, consent (e.g., not withdrawn), local pathology review, and PRC pathology review.
cancerHistory	Supports the creation, display, and editing of the cancer history of individual participants and their families
candidateRecord	Supports the display, recording, and editing of initial information on a case, gathered before a patient is accepted into a study
caseDw	Supports various operations for processing the CDR data into meaningful aggregates in the limited data warehouse for reporting
caseQMSignature	Supports the creation, display, and editing of the Quality Manager review signoff, indicating that the case met the quality standards
caseRecord	Supports the creation, display, and editing of information about individual cases
caseReportForm	Supports the aggregation of information about clinical cases for reporting
caseWithdraw	Supports the creation, display, and editing of information about an individual withdrawing consent for participation in the study. After withdrawal, a case is no longer cahubAcceptable, and it is not used for reporting or analysis.
chpBloodRecord	Supports the individual Collection Handling and Processing (CHP) study blood form, including creation, editing, and display
chpTissueRecord	Supports the individual CHP study tissue form, including creation, editing, and display

Domain Class	Description
`concomitantMedication`	Supports the creation, editing, and display of a portion of the Case Record covering the drugs taken by an individual
`ctcCrf`	Supports the creation, editing, and display of the Circulating Tumor Cells (CTC) study Case Report Form (CRF)
`deathCircumstances`	Supports the creation, editing, and display of a portion of the candidate record for the GTEx project, recording details around the death of the donor
`demographics`	Supports the creation, editing, and display of a portion of the candidate record having to do with the participant's social habits, such as alcohol and tobacco use
`deviation`	Supports the creation, editing, and display of information tracking reports of events that do not follow the required SOPs
`donorEligibilityGtex`	Supports the creation, editing, and display of information about the eligibility of an individual donor for GTEx, and if not eligible, the reasons why not
`feedback`	Supports PRC feedback and administrative review and response to the users
`feedbackFzn`	Supports PRC feedback and administrative review and response to the users, specifically regarding frozen tissue
`fileUpload`	Supports the uploading of files to the CDR in support of various records (e.g., case, specimen, deviation) and the display and download of those uploaded files as requested
`formMetadata`	Records meta data about an individual form, such as the SOP document number and the version upon which it is based
`generalMedicalHistory`	Supports the creation, editing, and display of information about an individual's prior medical treatments
`icdGtexNdri`	Supports the creation, editing, and display of consent forms for the GTEx experiment for the National Disease Research Interchange
`icdGtexRpci`	Supports the creation, editing, and display of consent forms for the GTEx experiment for the Roswell Park Cancer Institute
`icdGtexSc`	Supports the creation, editing, and display of consent forms for the GTEx experiment for the University City Science Center
`interviewRecord`	Supports the creation, editing, and display of information about an individual's interview during the individual's assessment for participation in the ELSI study

Domain Class	Description
`medicalHistory`	Supports the creation, editing, and display of specific details of an individual's medical history
`patientRecord`	Supports a special variation of patient information, used with CTC patients only
`prcForm`	Supports the creation, editing, and display of information from the caHUB PRC about a particular case
`prcReport`	Supports the creation, editing, and display of information from the caHUB PRC about a particular case for GTEx and BPV
`prcReportFzn`	Supports the creation, editing, and display of information from the caHUB PRC about a particular case for GTEx in which frozen tissue was collected
`prcSpecimenReport`	Supports the creation, editing, and display of information from the caHUB PRC about a particular specimen
`processingEvent`	Supports the creation, editing, and display of information regarding how things happened to a specimen, such as creating a slide or creating an image
`query`	Supports the creation, editing, and display of irregularities found in the data associated with a participant or with specimens. Queries are used to track problems that the data management team finds when reviewing source site entries so that the source site can correct them.
`serologyResult`	Supports the creation, editing, and display of information about a particular body fluid specimen
`shipDiscrepancy`	Supports the creation, editing, and display of information about problems with transferring specimens from one location to another. These might include delays in transfer, temperature out of range, or other problems spelled out in the shipping SOP.
`shippingEvent`	Supports the creation, editing, and display of information about transferring specimens from one location to another, including time, temperature, destination, and other information from the shipping SOP
`slideRecord`	Supports the creation, editing, and display of information about a mounted aliquot of a specimen
`sopVersion`	Supports the creation, editing, and display of tracking information about versions of individual SOPs. Because SOPs can change over time, the exact SOP in effect when collecting a specimen must be recorded in a consistent manner.
`specimenDw`	Supports various operations for processing the CDR data about specimens into meaningful aggregates in the limited data warehouse for reporting

Domain Class	Description
`specimenRecord`	Supports the creation, editing, and display of information about a specimen
`surgicalProcedures`	Supports the creation, editing, and display of information about the detailed surgical procedures used in tissue collection in BPV
`therapyRecord`	Supports the creation, editing, and display of information about a participant’s chemotherapy
`tissueRecoveryBms`	Supports the creation, editing, and display of information about the detailed tissue recovery in the biospecimen management system
`tissueRecoveryBrain`	Supports the creation, editing, and display of information about the detailed brain recovery in GTEx
`tissueRecoveryGtex`	Supports the creation, editing, and display of information about the tissue recovery in GTEx, including tissue fixation time
`trailAuditLog`	Supports the creation, editing, and display of information about updates, additions, and deletions to the CDR database
`user`	Supports the creation, editing, and display of information about individuals with access, at any level, to the CDR, including roles and privileges

The CDR Grails solution uses a Spring MVC Web application framework. Spring MVC is an extensible MVC, making it perfect for Grails. The Grails servlet extends Spring’s DispatcherServlet to bootstrap the Grails environment; then a single Spring MVC controller called org.codehaus.groovy.grails.web.servlet.mvc.SimpleGrailsController handles all Grails controller requests.

The SimpleGrailsController delegates to a class called org.codehaus.groovy.grails.web.servlet.mvc.SimpleGrailsControllerHelper that actually handles the request. This class breaks the handling of the request down into a number of steps. The entry point for the class is the handleUri method, which uses the following steps:

1. Parse the universal resource identifier (URI) into its components (controller name, action name, id, etc.).
2. Look up a GrailsControllerClass instance for the URI (see Figure 3).
3. Create a new Grails controller instance.
4. Configure the controller instance’s dynamic methods and properties.
5. Retrieve a scaffolder if scaffolding is enabled for the controller.
6. Get a reference to the closure action to execute for the URI.
7. Increment flash scope, moving the scope to its next state.
8. Get the view name for the URI.
9. Execute any interceptors that have been registered (e.g., Spring Security).
10. Execute the closure that is the controller action if the “before” interceptor did not return false.
11. Create a Spring MVC ModelAndView instance from the view name and the model returned by the closure action.
12. Execute any “after” interceptors registered, passing the returned model to the interceptor.

13. Return the Spring MVC ModelAndView instance.

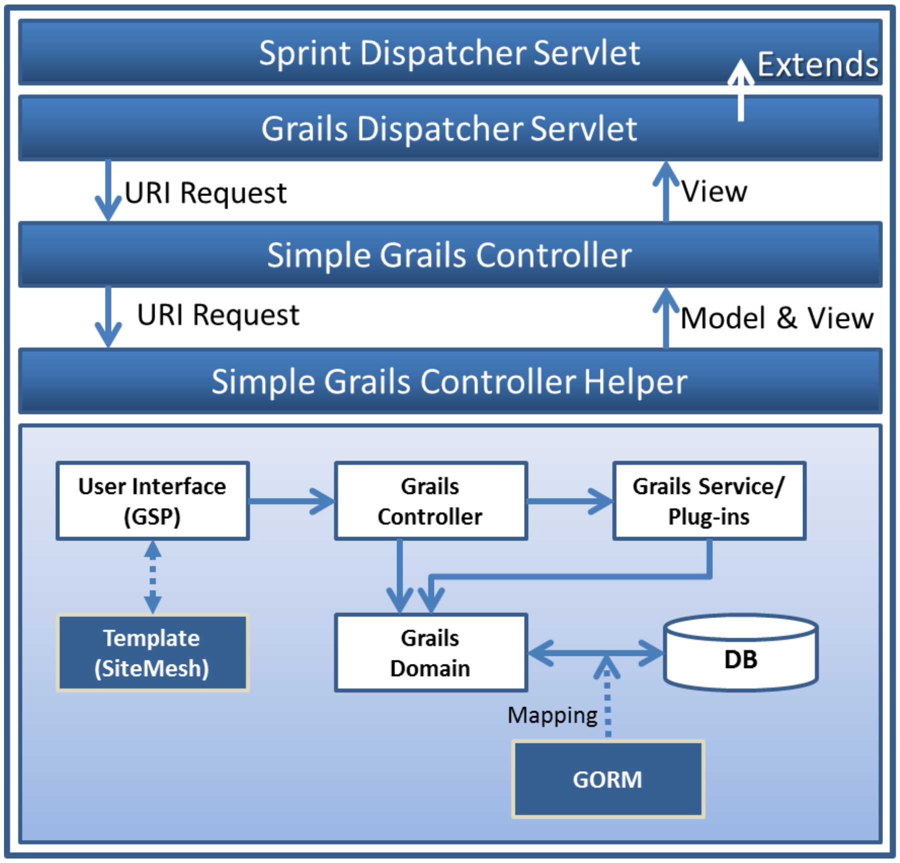

Figure 7: Grails Framework in Context of Spring MVC

CDR Grails is implemented as a standard Grails 2.2.4 project. CDR forms are implemented as Groovy Server Pages (GSPs); data records and static members are Grails domain classes (see Table 2); and XML interfaces are defined as Grails Services. Figure 8 through Figure 14 show the directory structure showing how the elements of the CDR map into Figure 7.

The configuration file, shown in Figure 3, controls the Spring Dispatcher Server from Figure 8. The restrictions that this configuration file imposes limit data access by both roles and controllers. External firewalls handle filtering by IP addresses.

Figure 8 provides a high-level view of the directory structure. This is basically an extension of the default Grails directory structure.

The paragraphs below describe the conf, controllers, domain, jobs, and services directories.

The `cahubdataservices/lib` directory contains JAR files for open-source utilities (shown in Table 3) used throughout CDR.

Table 3: JAR Libraries Used CDR

Library JAR Files	Description
`activation-1.1.jar`	The JavaBeans activation framework, used in processing mail
`mail-1.4.1.jar`	The Sun mail IMAPI protocol
`ojdbc6.jar`	The Oracle Java Database Connect utilities used for Oracle
`org.springframework.test-3.0.3.RELEASE.jar`	The test framework for Spring (not used)
`poi-3.9-20121203.jar`	The Apache Microsoft Document API for creating and manipulating Excel files for reporting

The `cahubdataservices/scripts` directory contains the file `_Events.groovy`, which is an event handler.

The `cahubdataservices/web-app` directory extends `cahubdataservices/grails-app/web-app` by linking in various JAR files found in other directories at this level.

```
    \---cahubdataservices
        +---grails-app
        |   +---conf
        |   +---controllers
        |   +---domain
        |   +---i18n
        |   +---jobs
        |   +---services
        |   +---taglib
        |   +---utils
        |   \---views
        +---lib
        +---scripts
        +---src
        \---web-app
```

Figure 8: High-Level View of CDR Directory Structure

The `conf` directory, shown in Figure 9, stores various files used in configuring the CDR and the Oracle data source. When needed, other configuration files go in this directory. Grails removes the need to add configuration in XML files for the CDR. Instead, the framework uses a set of conventions while inspecting the code of Grails-based applications. For example, a class name that ends with "Controller" (e.g., `LoginController`) and is in the `Controllers` folder is a controller. In general, the controllers related to base classes are stored in the `Controller` directory, shown in Figure 10.

```
         +---grails-app
         |   +---conf
         |   |   |   BootStrap.groovy
         |   |   |   BuildConfig.groovy
         |   |   |   Config.groovy
         |   |   |   DataSource.groovy
         |   |   |   DefaultQuartzConfig.groovy
         |   |   |   GrailsMelodyConfig.groovy
         |   |   |   QuartzBootStrap.groovy
         |   |   |   Searchable.groovy
         |   |   |   SecurityFilters.groovy
         |   |   |   UrlMappings.groovy
         |   |   |
         |   |   \---spring
         |   |           resources.groovy
```

Figure 9: CDR `conf` Directory Details

Figure 10 shows the `controllers` directory, which stores various files mapping between the database model and the views displayed in the Web interface. Grails uses controllers to implement the behavior of Web pages. For each CDR domain class, the associated controller file has the domain name concatenated with “Controller” and is of type .groovy. For example, the login controller is `LoginController.groovy`. Table 4 shows controllers unrelated to domain classes.

Table 4: Controllers Not Associated with Domain Classes

Controller	Description
`backoffice`	Supports monitoring of system functioning, activities, and RESTful services
`errors`	Supports the “Page not found” screen that appears when requested information is not available
`help`	Supports the system help screens that are available to users
`helpFileUpload`	Supports the upload of user manuals, SOPs, and other documents for the users
`home`	Supports the page that users see when they log in. This home page is a function of the individual user and their role.
`login`	Supports the authentication of users when they attempt to access the website
`logout`	Destroys the session
`userLogin`	Supports the reporting of user logins and the information stored in CDR about those users, such as affiliation

```
    \---cahubdataservices
        +---grails-app
        |   +---controllers
        |   |   \---nci
        |   |       \---obbr
        |   |           +---cahub
        |   |           |   +---datarecords
        |   |           |   |   \---qm
        |   |           |   +---datawarehouse
        |   |           |   +---forms
        |   |           |   |   +---bms
        |   |           |   |   +---bpv
        |   |           |   |   |   +---clinicaldataentry
        |   |           |   |   |   \---worksheet
        |   |           |   |   +---common
        |   |           |   |   |   \---withdraw
        |   |           |   |   \---gtex
        |   |           |   |       \---crf
        |   |           |   +---prc
        |   |           |   |   \---bpv
        |   |           |   +---prctumor
        |   |           |   +---register
        |   |           |   +---staticmembers
        |   |           |   \---util
        |   |           \---cahubdataservices
```

Figure 10: CDR `controllers` Directory Structure

The `grails-app/domain` directory, shown in Figure 11, stores the domain classes. Table 2 describes the individual domain classes. Each file is a .groovy file, and the file name is the name of the domain class. The directory structure divides the files into core functionality (e.g., authservice, data records, datawarehouse, ldacc, prc, prctumor, util) and forms (e.g., bms, bpv, common, gtex). Each study contains a mix of forms, some of which are common across multiple studies. The selection of forms for a study is defined by the needs of the study and is documented in the appropriate SOP. GORM manages all domain classes found in the domain. Methods are dynamically added to aid in persisting the class's instances. The files in Figure 11 map into the "Grails Domain" box in Figure 7.

```
    \---cahubdataservices
        +---grails-app
        |   +---domain
        |   |   \---nci
        |   |       \---obbr
        |   |           \---cahub
        |   |               +---authservice
        |   |               +---datarecords
        |   |               +---datawarehouse
        |   |               +---forms
        |   |               |   +---bms
        |   |               |   +---bpv
        |   |               |   |   +---clinicaldataentry
        |   |               |   |   \---worksheet
        |   |               |   +---common
        |   |               |   |   \---withdraw
        |   |               |   \---gtex
        |   |               |       \---crf
        |   |               +---ldacc
        |   |               +---prc
        |   |               |   \---bpv
        |   |               +---prctumor
        |   |               +---staticmembers
        |   |               \---util
```

Figure 11: CDR `domain` Directory Structure

Figure 12 shows a collection of directories defined in the Grails scaffolding.

The `grails-app/i18n` directory holds internationalization information for the website. This directory contains the file `messages.properties`, which contains settings for error messages used in data validation.

The `grails-app/jobs/cahubdataservices` directory contains a variety of Groovy utilities for performing periodic functions, as shown in Table 5.

Table 5: CDR Utility Job Classes

CDR Utility Job Classes	Description
`AgeUpdateJob.groovy`	Updates the age for GTEx (a calculated field)
`ExpirePasswordsJob.groovy`	Implements the NIH password expiration date
`LdaccImportJob.groovy`	Interfaces with the Broad Institute of MIT and Harvard to import results of molecular analysis
`ModuleTrackJob.groovy`	Implements tracking of BPV modules
`SpecimenDwJob.groovy`	Updates the internal data warehouse inside the CDR, used in generating real-time reports
`TextIndexJob.groovy`	Updates indexing of text fields for Lucene searching

The `grails-app/services/nci/obbr/cahub` directory is the repository for the CDR Grails service classes. A service class is a Plain Old Groovy Object (POGO), frequently with a name starting with a domain class and ending with “Service.groovy.” These service classes are Spring-loaded Groovy beans. Table 6 describes the CDR services that are not associated with a domain class.

Table 6: Service Classes Not Associated with a Single Domain Class

Non-Domain Class Service	Description
`AccessPrivilegeService`	Supports Spring in checking the access privileges of the user and role
`ActivityEventService`	Implements user email notification when various events take place (such as a case collected)
`BatchProcessingService`	Supports various periodic jobs run inside the CDR
`BmsService`	Service specific to the BMS study
`BpvCaseStatusService`	Service specific to the BPV study

Non-Domain Class Service	Description
`BpvElsiService`	Service specific to the ELSI study inside BPV
`CachedGtexDonorVarsExportService`	Service specific to the GTEx study
`ChpEventService`	Service specific to the BRN study
`ChpService`	Service specific to the BRN study
`CollectionEventService`	Service specific to tissue collection events
`GtexDonorVarsExportService`	Service specific to the GTEx study
`GtexTrfService`	Service specific to the GTEx study
`HubIdGenService`	Generates unique caHUB identifications
`LdaccService`	Interacts with external LDACC services
`MbbService`	Supports the University of Miami Brain Endowment Bank controller and views
`PrcTumorFormService`	Supports the PRC tumor report form
`ProcessingService`	Supports shipping and processing events as XML payloads when they are received from Van Andel via REST over HTTPS
`RpcService`	Supports Java remote procedure calls and RESTful calls
`SendMailService`	Supports email notification of users when triggers occur
`StaticMemberService`	Generically supports all CDR static members
`TextSearchService`	Supports the Lucene searches of records
`UtilService`	General utility service

The `grails-app/taglib/cahubdataservices` directory, shown in Table 7, contains Groovy code that dynamically generates the JavaScript associated with GSP tags in the HTML for the Web pages. This is part of the "User Interface" box in Figure 7.

Table 7: Dynamic HTML Classes

Classes for Implementing Dynamic Customized Code	Description
`BpvElsiSurveyLinkTagLib.groovy`	Implements customized GSP tags for the BPV ELSI study
`CaseRecordLinkTagLib.groovy`	Implements customized GSP tags for the Case Record display page
`JqueryDatePickerTagLib.groovy`	Implements a customized date-picker
`MedicationAdminTagLib.groovy`	Implements customized GSP tags for the BPV Medication form
`MultiselectTagLib.groovy`	Implements customized GSP tags for various data entry forms
`QueryTagLib.groovy`	Implements customized GSP tags for various data entry forms
`RadioButtonTagLib.groovy`	Implements customized GSP tags for various data entry forms
`WarningTagLib.groovy`	Implements customized GSP tags for various data entry forms

All contents of the directory `grails-app/util` are deprecated.

The `grails-app/view`, shown in Table 8, has one sub-directory for each domain class. Each entry under that directory is one or more GSP (.gsp) files. Each file's name describes a method in the corresponding controller class. Table 8 lists the typical methods that are automatically generated whenever you create a domain class (you can create other views and controller methods; the names *should*, by convention, match).

Table 8: Typical/Default Grails Server Pages

Grails Server Page	Description
`create.gsp`	Produces a UI to enter all values for a domain class and create a new instance
`edit.gsp`	Produces a UI to change the values of an instance of a domain class
`index.gsp`	The default method for a domain class, like `index.html`
`list.gsp`	Produces a list of persistent domain class objects
`show.gsp`	Produces a UI showing the contents of an instance of the domain class
`_form.gsp`	An include file with all the domain class attributes, used by the `create.gsp` and `edit.gsp` files.

```
    \---cahubdataservices
        +---grails-app
            +---i18n
            +---jobs
            |   \---cahubdataservices
            +---services
            |   \---nci
            |       \---obbr
            |           \---cahub
            +---taglib
            |   \---cahubdataservices
            +---utils
            \---views
```

Figure 12: CDR General Purpose Directories

Figure 13 shows the `cahubdataservices/src` directory, which contains Groovy, Java, and template code.

The `cahubdataservices/src/groovy/nci/obbr/cahub` directory contains one file, `CDRBaseClass.groovy`, which is the base class for all CDR Groovy classes. The CDR developers use it to create an abstract class that the domain classes have all extended. Extending the CDRBaseClass when implementing a domain class causes the domain class to inherit the "auditable" attribute, which logs all changes, inserts, updates (with old value and new value), and deletes, all with username and timestamp, in a special audit Log table.

The `cahubdataservices/src/groovy/nci/obbr/cahub/context` directory contains one file, `CDRApplicationEvent.groovy`, which responds to all Spring class events. The code in this file is an interceptor class that fires automatically whenever a Spring Security event (such as login or update to a domain class) is triggered.

The `cahubdataservices/src/groovy/nci/obbr/cahub/datarecords` directory contains one commonly used file, `DataRecordBaseClass.groovy`, which is the base class for all database records. DataRecordBaseClass extends CDRBaseClass, but it implements no new attributes.

The `cahubdataservices/src/groovy/nci/obbr/cahub/forms` directory contains one sub-directory for each study. Each study has a `FormBaseClass.groovy` file that is the base class for all forms used in that study's records.

The `cahubdataservices/src/groovy/nci/obbr/cahub/ldacc` directory contains one commonly used file, `Result.groovy`, which records the molecular results received daily from the Broad Institute of MIT and Harvard.

The `cahubdataservices/src/groovy/nci/obbr/cahub/security` directory contains files that customize the Spring Security model in Grails with CAS[5] authentication for single sign-on.

The `cahubdataservices/src/groovy/nci/obbr/cahub/staticmembers` directory contains one file, `StaticMemberBaseClass.groovy`, which serves as the base class for all CDR static classes.

The `cahubdataservices/src/groovy/nci/obbr/cahub/util` directory contains files for recording user access and event logging.

The `cahubdataservices/src/java` directory contains only one file, `GenCRC32.java`, which computes CRC 32 parity. The use of this file has been deprecated.

```
    \---cahubdataservices
        +---grails-app
        +---src
        |   +---groovy
        |   |   \---nci
        |   |       \---obbr
        |   |           \---cahub
        |   |               +---context
        |   |               +---datarecords
        |   |               +---forms
        |   |               |   +---bpv
        |   |               |   \---gtex
        |   |               +---ldacc
        |   |               +---security
        |   |               |   \---cas
        |   |               |       +---userdetails
        |   |               |       \---web
        |   |               +---staticmembers
        |   |               \---util
        |   |                   +---appaccess
        |   |                   \---pogo
```

5 https://grails.org/plugin/spring-security-cas

```
        |   +---java
        |   \---templates
```

Figure 13: CDR `src` Directory Structure

Figure 14 shows the `web-app` directory, which uses a standard web-app directory structure from J2EE. The standard `js` directory includes OpenSeadragon (Open Sea Dragon GitHub Repository, n.d.), used in visualizing pre-stored images of specimens. OpenSeadragon was released under the new BSD license. The rest of the directories under `web-app` contain custom JavaScript written for the CDR application or for the specific supported studies. The `WEB-INF` and `META-INF` directories are also located here, but they are not shown by default in the NetBeans IDE project view. `applicationContext.xml` is generated and managed by Grails, not the programmer.

```
    \---cahubdataservices
        +---grails-app
        \---web-app
            +---css
            +---images
            +---js
                +---bpv
                +---countdown
                +---ext
                |   \---vocab
                +---json
                +---openseadragon-bin-1.1.1
                +---prototype
                +---surveys
                \---timeentry
```

Figure 14: CDR `web-app` Directory Structure

Grails uses the concept of “convention over configuration.” This means that, typically, it uses the name and location of files instead of explicit configuration. Therefore, you will need to familiarize yourself with the directory structure provided by Grails 2.2.4.

`cdrdataservices` is the main application directory and contains the following sub-directories:

- `Configuration`: Contains Grails, Hibernate, and Spring configuration files and directories.
- `Controllers`: Holds the controller classes, the entry points into a Grails application. Grails subclasses Spring’s DispatcherServlet, which is used for delegating to CDR controllers.
- `Domain`: Holds the domain classes, which represent the persistent data for CDR, such as cases and specimens.
- `i18n`: Supports internationalization.
- `Scripts`: Holds Gant scripts.
- `Services`: Holds the server classes, which are Spring-managed beans.
- `taglib`: Contains GSP custom tag libraries.

- `utils`: Holds a variety of codec classes.[6]
- `Views`: Holds GSPs—the V (view) in MVC.

Figure 15 shows the file types (*.gsp, *.groovy, and *.jar) that are used in various layers of the Grails version of an MVC application.

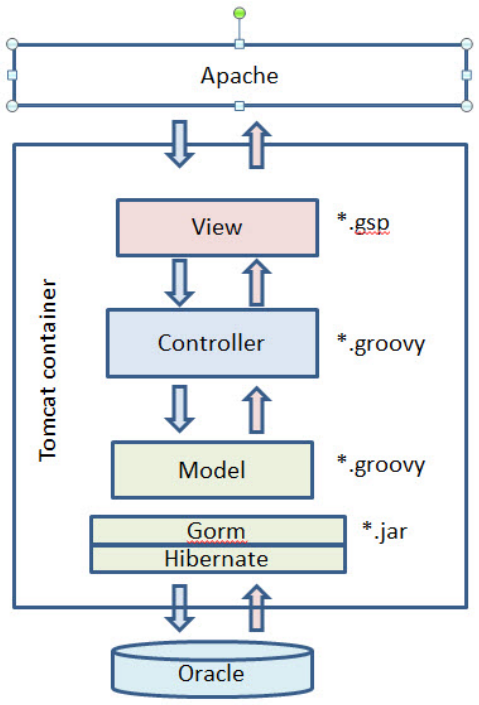

Figure 15: Grails Application in MVC Context and Server Aspect

4.4 Security Architecture

As shown in Figure 16 and the high-level diagram, the CDR is secured by Spring Security (formerly Acegi), using dependency injection and aspect-oriented programming (AOP). Every Web request for a resource (page) is filtered through Spring Security. Before each request is fulfilled, Spring Security decides whether the requesting user (human user or Web service) is (a) authenticated and (b) authorized.

The CDR has two modules, which are separate applications: the CDR-AR and the CDR-DS. Separating the modules separates the reporting (AR = “analytics and reporting”) from the data entry (DS = “data services”). This allows for easier deploying of new reports and analytics, without interrupting the users entering data, often around the clock. A single sign-on solution, CAS, was implemented for the two Web applications (which are on two different Tomcat servers, on two different VMs), so that users are not asked for their usernames and passwords when switching from the data entry module to the reporting module.

[6] See http://grails.org/doc/2.3.1/guide/single.html#codecs.

Spring Security supports the use of multiple containers for user authentication. NIH users can log in with their NIH credentials, and hospital (BSS) users can log in locally through Spring Security. Custom code was developed for the CDR to implement NIH account lifecycle and password policies for local users that mirror NIH policies. UI security and RESTful security are identical. REST calls are authenticated as local users (referred to by convention as “service accounts,” which is a Spring role), and both REST and UI users have stateful sessions.

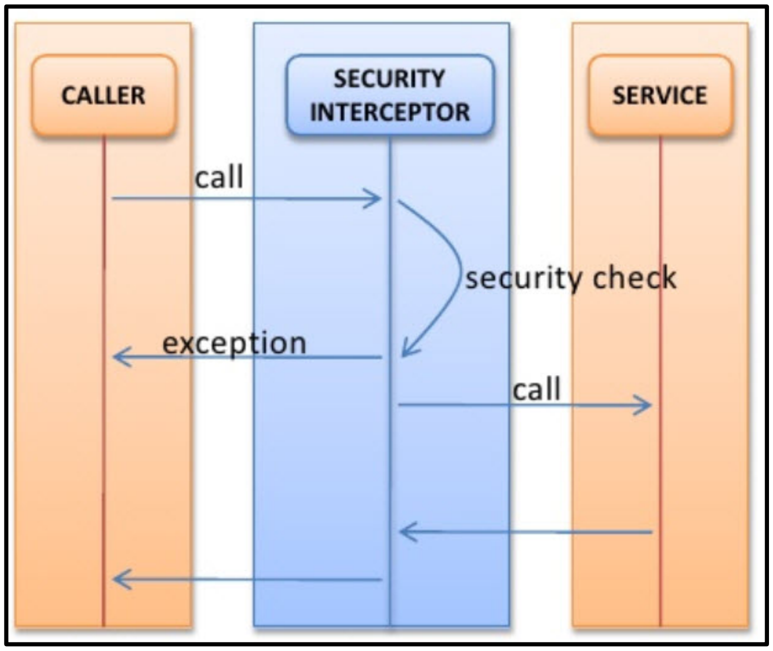

Figure 16: CDR Spring Security Model

The configuration of the Spring Security model is shown in Figure 3.

4.5 Communication Architecture

Communication with systems outside of the CDR happens in the following ways:

- Web services
- Email notifications
- Database CRUD operations

The following sections discuss these communications mechanisms.

4.5.1 Web Services

The CDR-DS invokes and accepts external communications by RESTful Web services. REST is an industry standard, built on top of the HTTP protocol as defined by Roy Thomas Fielding, Ph.D., using HTTP GET and POST. Data exchanged via the RESTful Web services are XML documents. Table 9 lists the RESTful services supported by the CDR.

Table 9: RESTful Services Available with CDR

RESTful Service Type	Purpose
Controlled Vocabulary	The CDR reports acceptable key words describing items in the study.
Shipping Event	BRIMS sends data to the CDR with shipping information and contents.

RESTful Service Type	Purpose
Collection Event	When a source site collects specimens, BRIMS sends the collection data to the CDR.
Processing Event	A processing event replaces the pre-note that was sent to notify the CDR of slides and images being processed from Van Andel.
PRC Summary Report on Entire Case	The CDR sends the pathology reports for each specimen and case to Van Andel and the Broad Institute.

4.5.2 Email Notifications

The CDR-DS sends email notifications using the industry-standard SMTP protocol. When the CDR detects various triggers, it sends notifications to pre-defined email groups. Members of the individual groups receive notifications at the same time. The text of each message describes what triggered the email, giving specifics (not containing PII or PHI) so that the recipients can take the appropriate action. Recipients are defined by Exchange distribution lists and in the application settings, which systems administrators can modify.

4.5.3 Database CRUD Operations

The CDR-DS and the CDR-AR use the proprietary communications protocol of the database vendor (Oracle) to execute CRUD operations on the dedicated, local instance of the database system. This layer is encapsulated by the GORM layer, as described in section 4.3, Software Architecture. There is no external direct access to the CDR database, nor does the CDR access any other database instances.

5 System Design

5.1 Business Requirements

SOP RM-0001, Requirements Management Procedures, describes the progression from a business need to a software feature. Appendix B describes this document's availability.

5.2 Database Design

Figure 17 shows a high-level overview of the database design.

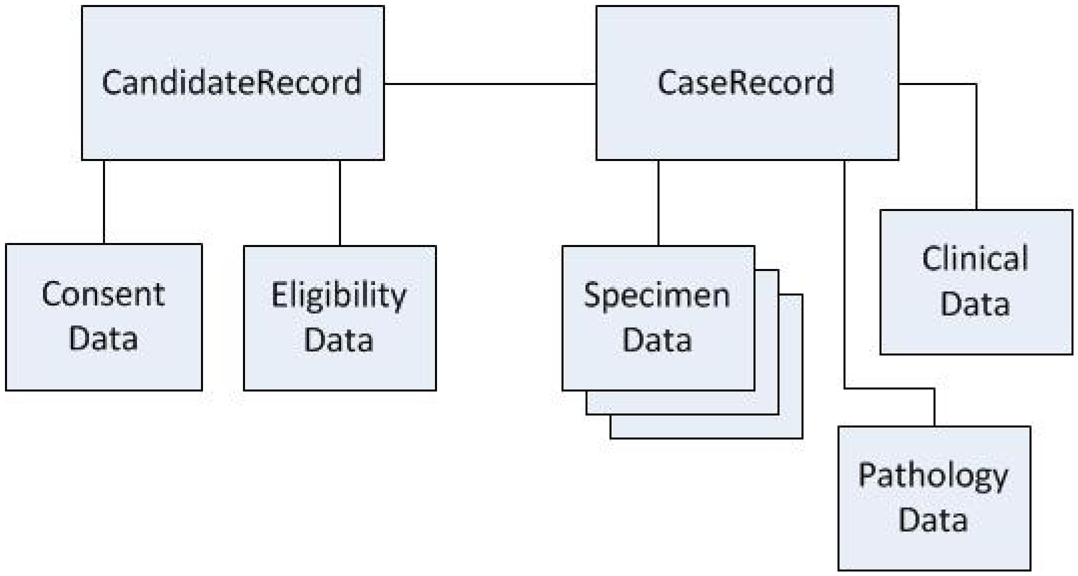

Figure 17: High-Level Overview of CDR Database Tables and Relations

A Candidate Record (domain class: CandidateRecord) contains the basic information about someone who may be involved with a study. Associated with the Candidate Record is the Consent Data, which contains the candidate's consent, or lack thereof, to participate in the study. The Eligibility Data records specific information about the candidate, including specifics of their disease and history. All of this information is used to determine whether the candidate meets the study's requirements (e.g., if the candidate's age is 65, but the study requires 20- to 40-year-olds, the candidate is ineligible). Details of consenting and eligibility vary greatly between studies, so these classes are implemented differently for each study.

For eligible candidates whose consent has been acquired, a Case Record (domain class: caseRecord) is created. The Case Record records specific information about the individual's involvement in the study. Associated with the case is Specimen Data (domain class: specimenRecord), which includes information about the collection and processing of the surgical products (e.g., blood, tissue). In all studies, a pathologist must review the specimens to confirm specimen type and quality; that review is recorded in the Pathology Data (domain class: prcReport). Clinical information related to the participant's care is recorded in the Clinical Data (domain class: clinicalDataEntry). Critical clinical information is study specific, so this class is often sub-classed, reflecting necessary details.

Grails uses GORM and Hibernate to automatically map between domain class objects and records in the underlying database. The table design, relations, and implementation details are automatically generated once the domain classes are defined.

Table 10 shows the mapping between tables in the database and domain classes.

Table 10: Mapping Between Database Tables and Domain Classes in the CDR

Database Table	Domain Class
CDRDS.ACCOUNT_ACTION	accountAction
CDRDS.ACCOUNT_STATUS	accountStatus
CDRDS.ACTIVITY_EVENT	activityEvent
CDRDS.APP_LISTING	applicationListing
CDRDS.APP_REQUEST	appRequest
CDRDS.APP_USERS	appUsers
CDRDS.AUDIT_LOG	Audit Log
CDRDS.DR_CANDIDATE	candidateRecord
CDRDS.DR_CASE	caseRecord
CDRDS.CASE_WITHDRAW	caseWithdraw
CDRDS.CASE_WITHDRAW_FORMS	caseWithdraw
CDRDS.DEVIATION	deviation
CDRDS.DR_CBR_API_EVENT	CbrApiEvent
CDRDS.DR_DERIVATIVE	derivativeRecord
CDRDS.DR_DERIVATIVE_SHIPPING_EVENT	Shipping events for derivatives
CDRDS.DR_DERIVATIVE_PROCESS_EVENTS	Processing events for derivatives
CDRDS.DW_CASE	case data warehouse
CDRDS.DW_SPECIMEN	Specimen data warehouse
CDRDS.FILE_UPLOAD	fileUpload
CDRDS.DR_GOAL	Accrual goals
CDRDS.DR_IMAGE	Image metadata
CDRDS.DR_KIT	kitRecord
CDRDS.DR_KIT_SHIP_EVENTS	Kit shipping events
CDRDS.DR_PATIENT	patientRecord (CTC only)
CDRDS.DR_PROCESSEVT	processingEvent
CDRDS.QUERY	Query
CDRDS.QUERY_ATTACHMENT	Query attachment
CDRDS.QUERY_RESPONSE	Query response
CDRDS.DR_SHIPEVT	shippingEvent
CDRDS.DR_SHIP_DISCREP	Shipment discrepancies
CDRDS.DR_SLIDE	Slide metadata
CDRDS.DR_SLIDE_SHIP_EVENTS	Slide shipment events
CDRDS.DR_SLIDE_PROCESS_EVENTS	Slide processing events
CDRDS.DR_SPECIMEN	specimenRecord
CDRDS.DR_SPECIMEN_DISCREPS	Specimen discrepancies
CDRDS.DR_SPECIMEN_PROCESS_EVENTS	Specimen processing events
CDRDS.DR_SPECIMEN_SHIP_EVENTS	Specimen shipment events

Figure 18 shows the tables in the CDR for basic information handling. When new forms are entered into the CDR, it automatically adds tables and fields through the Hibernate mechanism. This flexibility allows the data entered into the CDR to dynamically change values, adapting behavior, and meeting developing requirements.

For example, when the CDR is adapted for GTEx, the Oracle tables shown in Table 10 are generated. As the requirements for GTEx change—for example, because the CDR must support additional SOPs or because prior SOPs have been revised—this table structure changes automatically.

The detailed PDF files containing Figure 18 are available as documented in Appendix B.

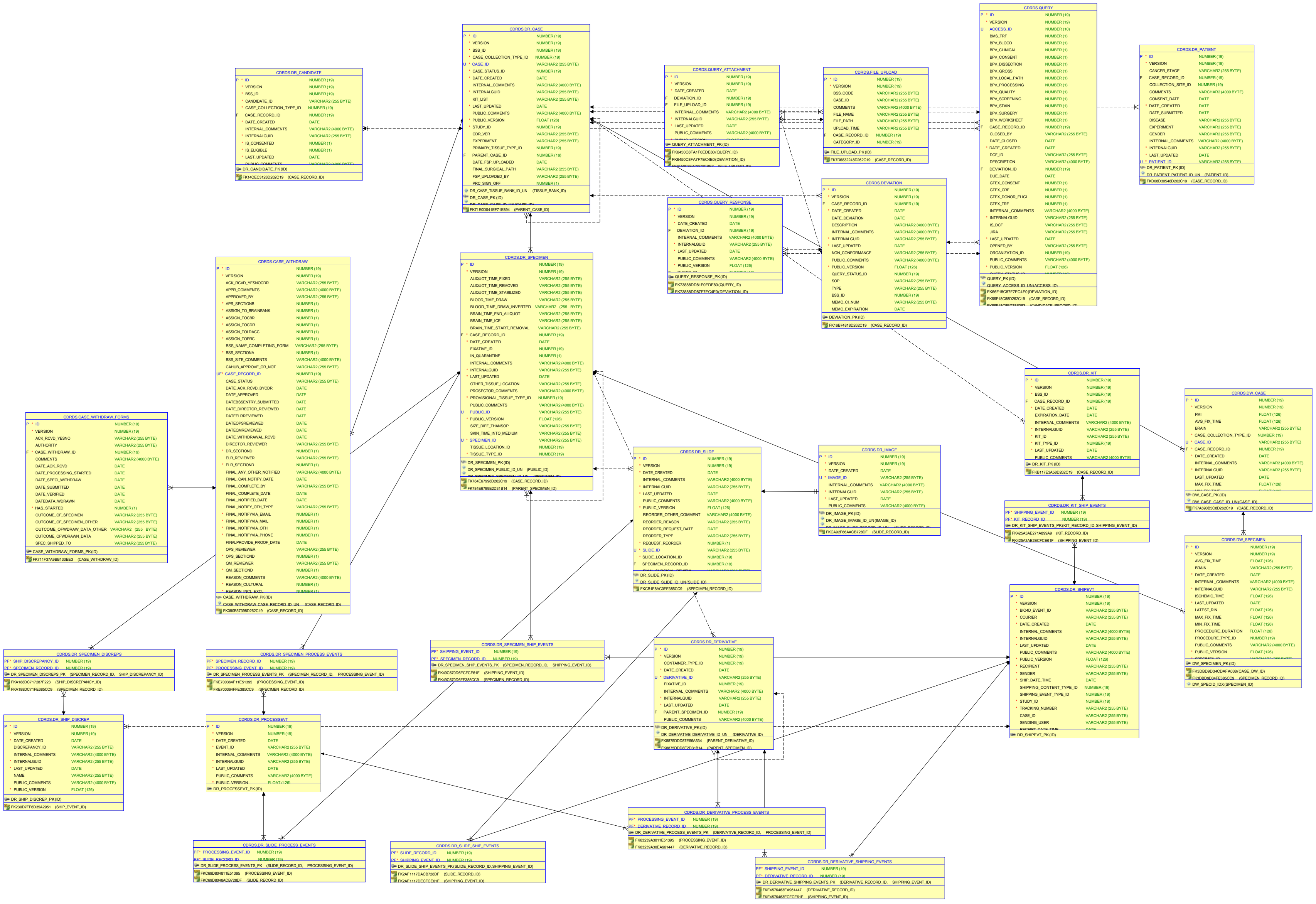

Figure 18: Study-Agnostic Tables for CDR

5.3 Data Conversion and De-Identification

The CDR does no data conversion. Users enter data in one or more electronic forms, and data are persisted in the database unchanged. Where applicable, input values are tested for acceptable ranges, either absolutely (e.g., the height of a person cannot be negative) or based on the values entered in other fields.

PHI may be entered in several of the forms managed by the CDR. The level of PHI stored is restricted via an LDS mechanism. The LDS data are stored in the underlying database. Access to areas of the CDR containing LDS data is controlled both by user entitlements and roles and by validation against Spring Security. If authorized to view the LDS data, full LDS data will be displayed on screens and reported; otherwise, data are de-identified through dynamic content redaction. Examples of de-identified elements include locations, birth dates, dates of procedures, and dates in relation to which procedures were performed or may be deduced are abstracted to years only. This dynamic redaction is performed on the fly by the server at the stage of rendering of Web service payloads or as HTML screens are generated.

Data may be aggregated in various ways for reporting and analytics. These aggregations may rely on or contain LDS data. Access to these reports is also validated against Spring Security.

5.4 Application Program Interfaces

The CDR-DS and CDR-AR code is documented independently in the CDR Groovy Code Documentation and the CDR CDI document. Appendix B describes the availability of both of these documents.

5.5 User Interface Design

The CDR's UI design is based on standard Web templating, using SiteMesh, scripting, and Cascading Style Sheets (CSS). HTML, JavaScript, and CSS are embedded in GSPs, which are rendered in the client with live data. The rendered output is a standards-compliant, cross-browser–compatible HTML page.

The UI requirements for how the CDR was to look to the BSS users were minimal. CDR users are familiar with paper forms, so the UI is designed to look and act like an electronic version of the paper artifacts. The primary UI includes tables, lists, and dynamic elements populated based on user input and response. The Web forms used to capture clinical data were based on the program SOPs for data capture. All users share the same style of interface, but fields and entire pages are restricted from some users. Each user field is Spring-loaded, so the capabilities of the user logged in are authenticated before display and before the user is allowed to enter or change any information.

5.5.1 Users, Roles, and Audiences

The CDR has to support different user types, roles, and privileges. Some users are external to the CDR, and some are internal. These users have either read-write or read-only access to the data. Some roles can see only certain aspects of the programs supported by the CDR, whereas others can see only data generated by their organization. Table 11 shows the roles and privileges that the CDR supports. Each role is configured and validated against Spring Security upon login. Users' privileges and access levels are determined by their roles and organizations.

Table 11: CDR's User Roles and Privileges

Role	Write	LDS (access to PII)	Global Access	Notes
BSS	Y	Y	N	Biospecimen source site staff role
DM	Y	Y	Y	Data manager
PRC	Y	N	Y	Pathology Resource Center pathologist
LDS	N	Y	Y	Read-only with access to HIPAA identifiers
R/O	N	N	Y	Basic read-only account
External Org	N	N	Y	External organization limited to a subset of read-only data
Super	Y	Y	Y	CDR super user account
Service (API)	Y	N	Y	Machine to machine accounts supporting Web Service APIs

Access to study-specific and functional areas of the CDR is available through a user's home page, as shown in Figure 19. Depending on their entitlements, users may see a different home page or be restricted to certain areas. Users with some power user roles, such as DM, LDS, and Super, have the ability to raise and lower their privileges as needed. Study areas (i.e., GTEx, BPV, BPV ELSI, and CTC projects) are displayed across the top. Functional areas (i.e., PRC, DM, Analytics & Reporting, and Vocabulary) are displayed on the second line.

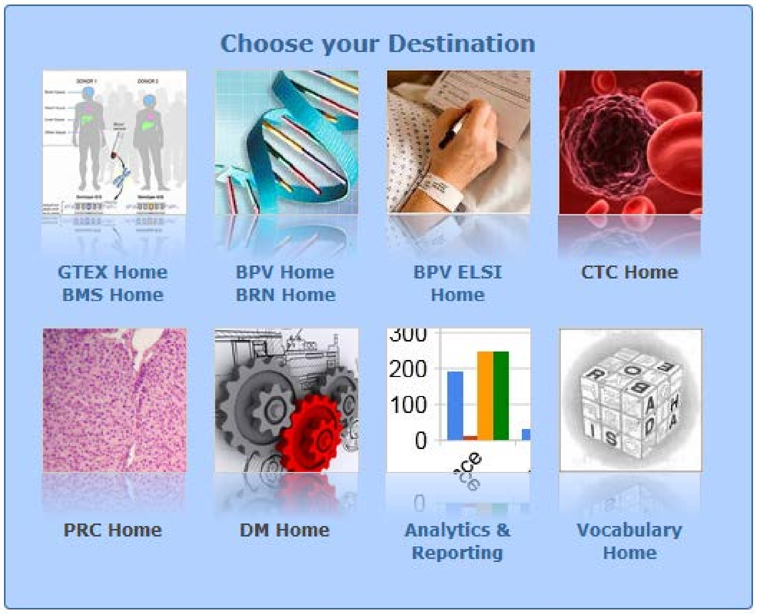

Figure 19: CDR Home Screen for BBRB and Leidos Users

5.5.2 Triggers

The CDR uses a mechanism called “triggers,” which send email messages to the appropriate users when some predefined event happens or a business rule is applied. Each trigger contains a number of elements, including code to check for the business logic to see if a matching event is happening, a pre-defined mail body text, and a list of email addresses for those who should be notified when the event occurs. Section 4.5.2 discusses the communications mechanism for emails. Individual triggers may be customized to include specific information in the body of the message (e.g., case ID, individual field values not containing PII or PHI).

T contains the full list of triggers and a description of the event associated with each trigger. When a trigger fires, it creates an SMTP message specific to the triggering event. That SMTP message hands off to an external mail server, which delivers the message to the appropriate experts. The users then take the appropriate action. Which users get the message depends on the SMTP mail list description; adding a person to a distribution list ensures that the person automatically gets all future messages for that list.

Table 12: Mail Distribution Lists for Various Triggers

Name of Trigger Distribution List	Triggering Event
APERIO_IMAGE_DISTRO	Notification: Whole-slide images are available at the CBR for a given study.
BPV_CASE_CREATION_DISTRO	Notification: A new BPV case was initialized at a BSS.
BPV_INFECTIOUS_DATA_DISTRO	Alert for follow-up: Data about infections were entered for a BPV case.
BSD	Berkeley Software Distribution
CASE_STATUS_CHANGE_DISTRO	Notification: A case status was changed by a BSS user or data manager.
COLLECTION_EVENT_DISTRO	Notification: A new GTEx case was collected at a BSS.
DM_FAST_TRACK_DISTRO	Notification: LDACC user requested priority data manager review.
FILE_UPLOAD_DISTRO	Notification: A new file attachment was uploaded to the CDR.
GTEX_CORE_UNACC_DISTRO	Alert for follow-up: A core GTEx tissue was marked unacceptable by PRC.
GTEX_DONELIGQ15YES_DISTRO	Alert for follow-up: TPM and data manager: Question 15 on Donor Eligibility as answered Yes.
INTERVIEW_STATUS_DISTRO	Notification: Interview status was changed for BPV-ELSI study.
LDACC_RIN_DISTRO	Notification: Report for nightly LDACC RIN transfer was called by the CDR.
NEW_INTERVIEW_DISTRO	Notification: New BPV-ELSI Interview was started at a BSS.
NEW_QUERY_TRACKER_DISTRO	Notification: A new query was created by Data Management.

Name of Trigger Distribution List	Triggering Event
PRC_ACTIVITY_DISTRO	Notification: General PRC CDR activity notifications for the PRC group
PRC_REPORT_AVAILABLE_DISTRO	Notification: PRC report is available for review.
PRC_REPORT_FLAGGED_QC_DISTRO	Notification: PRC report has been flagged for quality control review.
PROCESSING_EVENT_DISTRO	Notification: Specimens were processed at the CBR.
PROC_FEEDBACK_AVAILABLE_DISTRO	Notification: Procurement Feedback form is available for BSS review.
QT_RECUT_DISTRO	Notification: A slide recut has come through the query tracker.
SEPSIS_REPORT_DISTRO	Warning for follow-up by TPM: A GTEx case with sepsis information was entered into the CDR.
SHIPPING_EVENT_DISTRO	Notification: A specimen or kit shipment was sent by the CBR or BSS.
SHIPPING_RECEIPT_DISTRO	Notification: A specimen or kit shipment was received by the CBR or BSS.

5.6 CDR-AR, Analytics and Reporting

The CDR contains an analytics and reporting module (the CDR-AR) that analyzes real-time data, generating reports on demand for project and data managers. These reports are customized to the individual projects' needs, and new reports are easily deployed without causing any downtime for the CDR-DS or affecting data entry activity for the BSSs, PRC, or data management team. The CDR-AR is an ancillary Grails application on a Tomcat and VMware server which is distinct from those used by CDR-DS. It accesses the CDR-DS database in read-only mode and leverages the Google Charts API along with the open-source Java charting library Dynamic Reports.

You can view the CDR UI implementation in the study user guides made available to CDR users (see Appendix B) and in the clinical data capture artifacts in the program study SOPs.

5.7 System Performance

The CDR was built to support hundreds of users, 5 to 10 of whom may be using the system at any given time, across multiple scientific studies, both within and outside NCI, with 99.99% uptime. The CDR is used in three different U.S. time zones and around the clock for biospecimen procurement activities and clinical data management. Web service APIs are available at all times for pushing and pulling of data by collaborating institutions. Load testing and performance tuning was performed on the CDR, ensuring the availability of acceptable performance margins under heavy use.

Preliminary load test results indicated that the CDR can support 10 times the average usage levels (from 5 to 10 concurrent users) by the most strenuous user roles. Appendix B gives a reference to the latest CDR load test report.

5.8 Section 508 Compliance

Section 508 compliance testing is currently under way for the CDR. This section will be updated pending the findings.

APPENDIX A. KEY TERMS

The following table defines and explains terms and acronyms relevant to this document.

Term	Definition
ABCC	Advanced Biomedical Computer Center
AOP	Aspect-oriented programming: A programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns
BPV	Biospecimen Pre-analytical Variables: A study sponsored by the Biorepositories and Biospecimen Research Branch that used the CDR to manage study-specific data
BRIMS	Software system developed by the Van Andel Institute for automating processes related to biospecimen logistical data and metadata
BSS	Biospecimen source site: An institute from which human tissue is initially collected
CAS	Central Authentication Service, used to authenticate user by name and password
CDR	Comprehensive Data Resource
CDR-AR	Analytics and reporting module for the CDR
CDR-DS	Data services module for the CDR
CTC	Circulating Tumor Cells: A project scheduled to use the CDR
dbGaP	database of Genotypes and Phenotypes: A project of the National Center for Biotechnology Information that was developed to archive and distribute the results of studies that investigated the interaction of genotype and phenotype
DM	Data management: The people and activities preserving data integrity
ELSI	Ethical, Legal and Social Issues
FEA	Federal Enterprise Architecture
FNLCR	Frederick National Laboratory for Cancer Research
GORM	Grails Object Relational Mapping
Grails	A powerful computer software framework based on the Groovy programming language and emphasizing rapid software development of Web-based applications
GTEx	Genotype-Tissue Expression program: A project of the NIH Common Fund

Term	Definition
LDACC	Laboratory, Data Analysis, and Coordinating Center, currently run by the Broad Institute
LDS	Limited data set: A reflection of the central data where PHI and PII have been protected
LIMS	Laboratory information management system
PHI	Protected health information: Health information, including demographic information, that relates to an individual’s physical or mental health or the provision of or payment for health care
PII	Personally identifiable information: Individually identifiable health information
RESTful	A type of Internet service standard interface, typically between programs exchanging information
SOP	Standard operating procedure: A detailed document precisely describing the performance of a protocol
UI	User interface: The Web-based graphical interface that enables various groups to enter and retrieve data from the CDR-DS or the CDR-AR
XML	Extensible Markup Language

APPENDIX B. REFERENCES

The following table summarizes the documents referenced in this document.

Document Name	Description	Location
GTEx CDR User Guide	Describes how the CDR should be used with the GTEx project	IT-9006-GTEx_CDR_Data_Services_Users_Guide_v3.01.docx
BPV CDR User Guide	Describes how the CDR should be used with the BPV project	IT-9004_-_caHUB_CDR_User_Guide_for_BPV_101012.pdf
Requirements Management Procedures	Provides instruction and guidance for caHUB requirements management, including information on identifying, reviewing, approving, and documenting new or updated requirements	SOP RM-0001 Requirements Management Procedures
GX3-OP-9001	Provides a detailed description of GTEx (and CDR) work flows	GTEx Project MasterControl documentation
Google Charts API	Allows an individual to easily create a chart from data, embedding it in a Web page	https://developers.google.com/chart
CDR Groovy Code Documentation	Provides an automatically generated HTML index of the code making up the CDR and CDR-AR structure	GitHub repository
caHUB Interface Control to CDR	Describes the mechanism that the CDR uses to communicate to caHUB resources	caHUB Interface Control Document.pdf, found in the GitHub repository
CDR Load Test Results	Provides results of load testing of CDR	CDR Load Test.pdf, found in the GitHub repository
Cross Mapping	Maps the caHUB concept to each of the caHUB terms, study forms, Field Labels (on the CDR form), GTEx de-identified exports, CDR Databases (table and column) to CDR key and value	caHUB to CDR Mapping.xlsx found in the GitHub repository
CDR-DS Data Model	Shows the data structures and relations; automatically generated from the Oracle server fields	GTEx Project MasterControl documentation

APPENDIX C. GTEX SYSTEMS DESCRIPTIONS

The following table provides high-level definitions of the systems used within the GTEx workflow that track the chain of custody of the collected specimens and data. These systems are referenced in the workflow steps.

System	Location	Potential System	Function
Biospecimen Inventory Management System	BSS	N/A	Used only within each BSS, the Biospecimen Inventory Management System (BIMS) is the system that stores inventory information about specimens collected, including the relationship between the patient's medical records, Donor/Patient ID, and Case ID. It links to the specimens stored at the caHUB CBR and the associated data in the CDR using the Case ID.
Biospecimen Inventory Management System	CBR	Bio4D + spreadsheets	Accessed through the caHUB's Comprehensive Biospecimen Resource (CBR), this BIMS refers generically to the system that stores inventory information about specimens collected at the BSSs and stored/processed by the CBR. This information includes the initial contents and labels on kits and shipment tracking information. All inventory and processing data regarding the specimens is initially referenced using the Specimen IDs.
Image Management System	CBR	Aperio	Accessed through the caHUB's CBR, this system stores information about the digital images made from specimen/sample slides, all referenced using the Case/Specimen/Sample ID.
Clinical Data System	CDR	CDR-DS	Accessed through the CDR, the interim caTissue system is a Web portal and stores all BSS case collection, handling, inventory, and processing data as well as clinical, pathology, and molecular data referenced using the BSS Case/Specimen/Sample/HUB-ID. This system includes the ID Generation Service that will take Specimen/Sample IDs and generate, print, and store corresponding Case IDs.

System	Location	Potential System	Function
Pathology Review System	CDR	CDR-DS	Accessed through the CDR, this notional system compiles the comments and results from the pathology evaluation performed by the PRC. Images from Aperio imaging systems are used in the pathology review but stored at the project's CBR.
Molecular Review System	CDR	CDR-DS	Accessed through the CDR, this notional system will compile the annotations and results from the CBR's molecular review.

APPENDIX D. GTEX STANDARD OPERATING PROCEDURES

The following is a list of the currently published Leidos Biomedical Research/Office of Biorepositories and Biospecimens Research SOPs used for the GTEx project.

SOP Number	Title
GX.0001	GTEx Beta Phase PM Collection of Normal Tissues
GX.0002	GTEx Kit Receipt Supply Shipping Instructions
GX.0003	GTEx caTissue Data Entry SOP
GX.0004	GTEx Data Management SOP
GX.0005	GTEx Beta Assessment Metrics PM Collection for Normal Tissues
GX.0006	GTEx Case Report Form Completion SOP
GX.0007	GTEx Chain of Custody SOP
QM.0004	Nonconformance Corrective and Preventive Action Procedure
SS.0001	BSS Training Procedure
SS.0002	GTEx BSS Beta Assessment Metrics Postmortem Collection of Normal Tissues Procedure

APPENDIX E. COMPREHENSIVE DATA RESOURCE DESIGN APPROVAL

The undersigned acknowledge that they have reviewed the document and agree with the information presented therein. Changes to this document will be coordinated with and approved by the undersigned or their designated representatives.

Signature:	______________________	Date:	__________
Print Name:	______________________		
Title:	______________________		
Role:	______________________		

Signature:	______________________	Date:	__________
Print Name:	______________________		
Title:	______________________		
Role:	______________________		

Signature:	______________________	Date:	__________
Print Name:	______________________		
Title:	______________________		
Role:	______________________		