*Comprehensive Data Resource Lite (CDR-Lite*) Design

Version 0.2

17-Mar-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 1.0 of the CDR Lite.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version Number** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Description of Change** |
| 0.2 | Dave Tabor | 11-Mar-2016 |  |  | First Draft - CDR Lite. |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

TABLE OF CONTENTS

[1 Introduction 5](#_Toc445989273)

[1.1 Purpose of the CDR-Lite Design Document 6](#_Toc445989274)

[2 High level view of CDR-Lite 6](#_Toc445989275)

[2.1 CDR-Lite Capabilities 6](#_Toc445989276)

[2.2 Using the CDR-Lite 6](#_Toc445989279)

[3 General Overview and Design Guidelines/Approach 7](#_Toc445989281)

[3.1 Assumptions / Constraints / Standards 7](#_Toc445989282)

[4 Architecture Design 8](#_Toc445989285)

[4.1 Logical View 8](#_Toc445989286)

[4.2 Hardware Architecture 8](#_Toc445989292)

[4.3 Software Architecture 9](#_Toc445989293)

[4.4 CDR-Lite Domain Class Model 10](#_Toc445989294)

[4.5 Security Architecture 23](#_Toc445989295)

[4.6 Communication Architecture 24](#_Toc445989296)

[4.6.1 Web Services 24](#_Toc445989297)

[4.6.2 Email notifications 24](#_Toc445989301)

[4.6.3 Database CRUD operations 25](#_Toc445989302)

[5 System Design 26](#_Toc445989303)

[5.1 Database Design 26](#_Toc445989306)

[5.2 Data Conversion and DE-Identification 30](#_Toc445989307)

[5.3 User Interface Design 30](#_Toc445989312)

[5.3.1 Users, Roles and Audiences 30](#_Toc445989313)

[5.3.2 Triggers 32](#_Toc445989314)

[Appendix A. Key Terms 34](#_Toc445989323)

[Appendix B. CDR-Lite Design Approval 40](#_Toc445989324)

**Table of Figures**

[Figure 1- Virtual Machine Network Architecture 9](#_Toc445989325)

[Figure 2- Current VM Configurations 9](#_Toc445989326)

[Figure 3- Grails Framework Architecture 10](#_Toc445989327)

[Figure 4- Grails framework in context of Spring MVC 14](#_Toc445989328)

[Figure 5 - High Level Overall View of CDR-Lite Directory Structure 16](#_Toc445989329)

[Figure 6 - CDR-Lite conf Directory Details 17](#_Toc445989330)

[Figure 7 - CDR-LITE controllers Directory Structure 18](#_Toc445989331)

[Figure 8 - CDR-Lite domain Directory Structure 18](#_Toc445989332)

[Figure 9 - CDR-Lite General Purpose Directories 20](#_Toc445989333)

[Figure 10 - CDR-Lite src Directory Structure 22](#_Toc445989334)

[Figure 11 - CDR-Lite web-app directory Structure 22](#_Toc445989335)

[Figure 12 - Grails Application in MVC Context and Server Aspect 23](#_Toc445989336)

[**Figure 13- CDR-Lite Spring Security Model** 24](#_Toc445989337)

[Figure 14- High-Level Overview of CDR Lite Tables and relations 26](#_Toc445989338)

[Figure 15- CDR-Lite’s User Role and privilege table 31](#_Toc445989339)

[Figure 16- Example CDR-Lite Home Page 32](file:///C:\Users\landerwb\Documents\caHUB\CDR%20Lite\CDR-Lite%20Architecture\CDR%20Lite%20architecture%2017-Mar-2016%20-%20wbl.docx#_Toc445989340)

[Figure 17- CDR Home Screen for BBRB and Leidos Users 32](file:///C:\Users\landerwb\Documents\caHUB\CDR%20Lite\CDR-Lite%20Architecture\CDR%20Lite%20architecture%2017-Mar-2016%20-%20wbl.docx#_Toc445989341)

[Figure 18- Mail Distribution List for Various Triggers 33](#_Toc445989342)

**Table of Tables**

[Table 1- CDR-Lite Domain Classes 10](#_Toc445989343)

[Table 2 - Spring Security Configuration File Config.groovy, Showing Roles and Restrictions 15](#_Toc445989344)

[Table 3 - Controllers Not Associated with Domain Classes 17](#_Toc445989345)

[Table 4 - CDR-Lite Quartz Scheduler Jobs 19](#_Toc445989346)

[Table 5 - Service Classes Not Associated with a Single Domain Class 19](#_Toc445989347)

[Table 6 – Custom Groovy Tags 20](#_Toc445989348)

[Table 7 - Typical/Default Grails Server Pages 20](#_Toc445989349)

[Table 8 - RESTful Services Available with CDR-Lite 24](#_Toc445989350)

[Table 9 - Mapping between Database Tables and Domain Classes in CDR 27](#_Toc445989351)

# Introduction

CDR-Lite is an open source extension of the CDR, intended for researchers customization in recording the metadata around specimen/tissue creation, processing, and storage.

The Comprehensive Data Resource (CDR) was developed to meet the challenges of real-world data collection of information about tissues gathered in the early stages of the Biospecimen Lifecycle[[1]](#footnote-2) . This includes collection information about potential candidates, their eligibility criteria and consent, medical/surgical procedures used, acquisition, handling, processing, and storage. As the focus for biospecimen-based studies for cancer have turned to the molecular level, it is more important than ever that more stringent specimen annotation techniques are applied in the lifecycle.

The initial design of CDR reflects the changing requirements of real world collections.

* As the collection process progresses, the need for different, or more detailed information may present itself. This changing need requires rapid changes (on the order of days) in forms, web pages, work flows, and database structure.
* Various review cycles must be supported. Reviews for data entry or processing errors (Data Management), specimen quality (both internal and external Pathology Review) must be supported.
* Geographic separation between those performing on the supported projects must be supported. The collection and processing sites may change dynamically during the course of the project.
* The Interface needs to reflect the historical paper-based forms process, to minimize the learning curve for the variety of users.

The first project producing requirements for CDR was the NIH Common Fund’s Genotype-Tissue Expression (GTEx) program[[2]](#footnote-3). GTEx collected 961 cases, with over 24,000 individual specimens.

The CDR code was later extended to cover a second project, Biospecimen Pre-analytical Variables (BPV)[[3]](#footnote-4). BPV collected information on over 300 cases, with specimens covering four different cancer types (and associated normal tissue), from four different tissue source-sites (Hospitals and University Medical Centers).

Following the successful execution of the BPV, GTEx, and other projects the decision was made to extract the features of CDR that might lend themselves more generally to other studies. The purpose and intent of the CDR-Lite is so that researchers can take advantage of the best practices of biospecimen collection that have been worked out over the course of several years, and with much expert consultation, and apply the software in their own environments, without a lengthy development cycle.

## Purpose of the CDR-Lite Design Document

The CDR-Lite Design Document provides the necessary information to describe the architecture. This gives the development team guidance on architecture of the CDR-Lite if 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) may on occasion be shared with the client/user, and other stakeholders whose input/approval into the UI is needed.

# High level view of CDR-Lite

## CDR-Lite Capabilities

The CDR-Lite is a web application, custom built to support specimen collection, clinical data entry, specimen logistics, as well as curation of study data. The capabilities reflect the best practices that have been determined over the years for a good out-of-the-box study management system, and include:

* Allowing remote users (e.g., researchers, clinicians, and support staff) to securely enter, revise, and review data about biospecimen collection through a standard (HTTPS) web interface via a series of electronic forms with a sophisticated role-driven workflow.
* Triggering responses, automatically communicating timely information to project managers and data analysts
* Assisting Quality Assurance by auditing process flows through Data Management and Pathology teams
* Controlling display of PII based on user entitlements and roles

## Using the CDR-Lite

Users are provided an account based on predefined study roles.

Setting up a study does not require custom coding and configuration. The CDR-Lite’s core data model is robust and can be extended to accommodate new studies. The level of effort needed in making changes to the CDR-lite will vary with a new project’s requirements.

Grails was chosen as the implementation language, because it reduced time needed in development by supporting automated generation of both web interfaces and automated management of the database schema, persistence, and searching. No DBAs are needed once the supporting Oracle or PostgreSQL instance is up and available; Changes in the schema are managed internally by GORM. All database access is based on Hibernate 3, which manages table definition updates, as well as content updates.

# General Overview and Design Guidelines/Approach

This section describes the principles and strategies to be used as guidelines when modifying the system.

## Assumptions / Constraints / Standards

The CDR-Lite architecture is designed for maximum flexibility in meeting changing and evolving requirements. As such, it is constrained to use standard services (e.g., RESTful web service interfaces, XML, and JSON data exchanges).

The CDR-Lite is designed to be HIPAA compliant with a Limited Data Set[[4]](#footnote-5). In the government arena, this greatly simplifies such things as security approval and FISMA compliance. 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, there are a variety of roles which may be assigned to a user, which limit their ability for accessing sensitive information. This is discussed in detail in section 4.5

# Architecture Design

The CDR-Lite is an enterprise-level application built around the motto “Science First.” The technologies chosen to build the CDR-Lite facilitated a faster development time through rapid and agile software methods. The open source Grails framework was utilized along with other proven open source technologies.

## Logical View

The CDR-Lite provides secure user access to case and biospecimen sample data based on pre-defined roles and privileges. Personally Identifiable Information (PII) and Protected Health Information (PHI) are restricted to a limited data set (LDS), with access given only to authorized users, through dynamic content redaction. Intuitive graphic user interfaces for the Biospecimen Source Sites (BSS) streamline data entry workflow by strictly following SOPs for sample collection and processing. Contextual automated data checks and business rule validations confirm data integrity and biospecimen collection/preservation SOP adherence.

Users of the web forms interface at collaborating institutions are granted access through application-specific user accounts performing data entry and accessing existing data entered under their institutions activities. Roles for BSS, Administrative and Data Manager are assigned as needed

## Hardware Architecture

The CDR-Lite is currently hosted on two virtual machines located at the Frederick National Laboratory for Cancer Research (FNLCR). As shown in Figure 1, separate virtual machines are used to host the CDR-Lite and the Apache front-end. This architecture was chosen as a best practice. In a typical web application deployment, the Apache server would be isolated in the DMZ with the Tomcat and database assets behind the firewall.

The SMTP Mail server is a shared mail server for all of NIH, and is not dedicated to CDR-Lite activities. Mail alert messages are sent to various mail groups when associated events are detected in CDR-Lite. The messaging discussion continues in sections 4.6.2 and 5.3.2.

Figure 2 shows the current virtual machine configurations for the CDR-Lite.

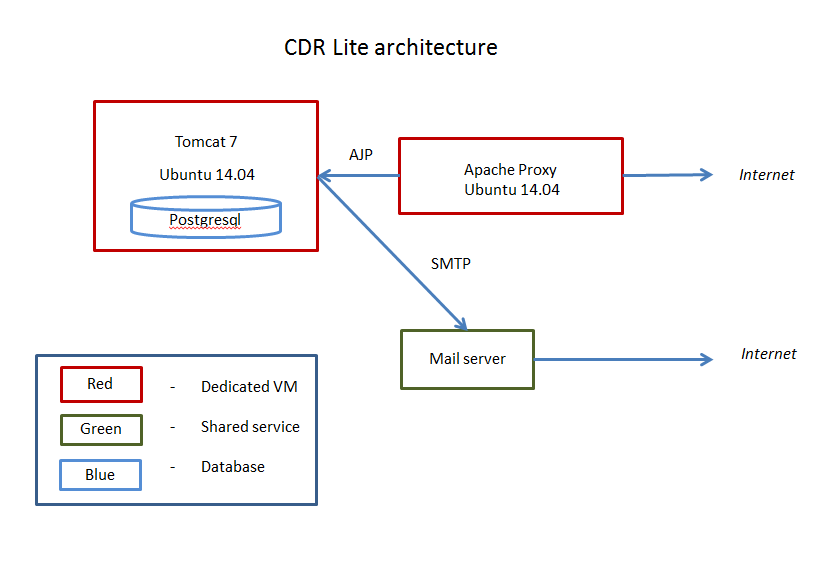


Figure 1- Virtual Machine Network Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| **VMWare ESX Host** | **RAM** | **CPU** | **Disk storage capacity** |
| Tomcat | 4096 MB | 2 - Westmere Xeon Core i7 | 68 GB |
| Apache | 2048 MB | 1 - Westmere Xeon Core i7 | 15 GB |

Figure 2- Current VM Configurations

## Software Architecture

Grails provides the software architecture as a framework shown in Figure 3 and Figure 12. The definition of Grails is maintained at <https://grails.org/> . The compiled Grails language (Groovy) runs, like other Java languages, in a Java Virtual Machine. For web applications, this Java Virtual Machine is shared with the Tomcat application container. Spring supports elements of the enterprise service layer and interactive user interface, the Grails Object Relational Mapping (GORM). SiteMesh is a decorator engine, supporting view layouts in generating web page displays, and the web services maps various web actions to code specific for those events.

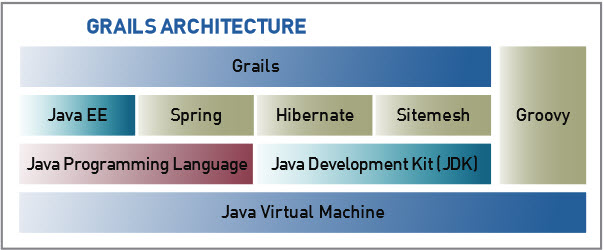


Figure 3- Grails Framework Architecture

## CDR-Lite Domain Class Model

The heart of the CDR-Lite is its domain model, that is, 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 3, 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 MVC) is the same as the corresponding Controller and Views. The location of the file in the CDR-Lite 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 application are already available. When a domain class is defined in NetBeans, right click on it, and choose “generate all.” Grails generates the Controller and default views for your domain class.

Table 1 enumerates the domain classes making up the CDR-Lite, and gives a brief description.

Table 1- CDR-Lite Domain Classes

|  |  |
| --- | --- |
| **Package name** | |
| cdrlite |  |
| **Domain Classes** | **Description** |
| ErrorsController.groovy | Customizable error handler generated by grails. Not modified for CDR Lite |
| **Package name** |  |
| nci\bbrb\cdr\authservice |  |
| **Domain Classes** |  |
| CdrRole.groovy  CdrUser.groovy  CdrUserRole.groovy | CDR Users and Roles generated by Spring Security plugin and renamed to “CdrUser, CdrRole and CdrUserRole.” In Oracle, User is a reserved word and using it as a table name resulted in troubles. |
| **Package name** |  |
| nci\bbrb\cdr\datarecords |  |
| **Domain Classes** |  |
| CandidateRecord.groovy | Candidate Record |
| CaseRecord.groovy | Case Record |
| ImageRecord.groovy | Image Record (for whole slide images) |
| PhotoRecord.groovy | Photo Record (for Gross Tissue preparation) |
| ProcessingEvent.groovy | Processing event |
| SlideRecord.groovy | Slide Record |
| SpecimenRecord.groovy | Specimen Record |
| **Package name** |  |
| nci\bbrb\cdr\forms | Forms package for CDR Lite |
| **Domain Classes** |  |
| CancerHistory.groovy | Cancer History form |
| ClinicalDataEntry.groovy | Clinical Data Entry form |
| ConsentVerification.groovy | Consent Verification form |
| Demographics.groovy | Demographics form |
| GeneralMedicalHistory.groovy | General Medical History form |
| HealthHistory.groovy | Health History form |
| MedicationHistory.groovy | Medication History form |
| ScreeningEnrollment.groovy | Screening and Enrollment form |
| SlidePrep.groovy | Slide Prep form |
| SlideSection.groovy | Slide Sectioning and Staining form |
| SocialHistory.groovy | Social History form |
| SurgeryAnesthesia.groovy | Surgery Anesthesia form |
| TherapyRecord.groovy | Therapy Record form |
| TissueGrossEvaluation.groovy | Tissue Gross Evaluation form |
| TissueProcessEmbed.groovy | Tissue Processing and Embedding form |
| TissueReceiptDissection.groovy | Tissue Receipt and Dissection form |
| **Package name** |  |
| nci\bbrb\cdr\forms\blood | A special package for the blood form |
| **Domain Classes** |  |
| Aliquot.groovy | Blood Aliquots |
| Blood.groovy | The main Blood form |
| CollectionTube.groovy | Collection Tubes |
| Draw.groovy | Blood Draws |
|  |  |
| **Package name** |  |
| nci\bbrb\cdr\prc | PRC Reports |
| **Domain Classes** |  |
| PrcReport.groovy | The PRC Report form |
| **Package name** |  |
| domain\nci\bbrb\cdr\staticmembers | Static members are where we store, maintain and modify “controlled vocabulary” items. These are typically, but not always, presented to the user in forms as a drop-down list. The contents can be modified in the Back Office (available only to Admins) |
| **Domain Classes** |  |
| ActivityType.groovy | Various Activity Types can trigger events, such as sending an Alert e-mail |
| BloodAliquotType.groovy | Used for the Blood form |
| BloodCollectionReason.groovy | Used for the Blood form |
| BloodDrawTech.groovy | Used for the Blood form |
| BloodDrawType.groovy | Used for the Blood form |
| BloodTubeType.groovy | Used for the Blood form |
| BSS.groovy | Static list of BSSs |
| CaseAttachmentType.groovy | Attachment Types for files uploaded and attached to a Case |
| CaseCollectionType.groovy | Collection Type |
| CaseStatus.groovy | Case Status |
| ContainerType.groovy | Container Type for Specimen collection |
| Fixative.groovy | Fixative for Specimen collection |
| Organization.groovy | Organization. Some organizations are BSSs |
| PrcAcceptability.groovy | PRC Acceptability status |
| QueryStatus.groovy | Query Status for the Query Tracker |
| QueryType.groovy | Query Type |
| StorageTemp.groovy | Storage Temperature |
| Study.groovy | Study |
| TissueCategory.groovy | Tissue Category |
| TissueLocation.groovy | Tissue Location |
| TissueType.groovy | Tissue Type |
| **Package name** |  |
| nci\bbrb\cdr\util | Utility Domain classes for CDR Lite |
| **Domain Classes** |  |
| ActivityEvent.groovy | Holds a record of Activities that Trigger Events |
| AppSetting.groovy | Dynamic Application Settings that contain eg: the Login Bulletin message to display, lists of users who can receive e-mail event notifications, etc. |
| FileUpload.groovy | Files uploaded and their path on the server |
| UserLogin.groovy | Record of Users logged in and login history. |
| **Package name** |  |
| nci\bbrb\cdr\util\querytracker |  |
| **Domain Classes** |  |
| Deviation.groovy | Used to record deviations from approved SOPs |
| Memo.groovy | Memos attached to a case record approved changes |
| Query.groovy | Query records of data management activities resolving data discrepancy issues |
| QueryAttachment.groovy | File attachments to a query |
| QueryResponse.groovy | BSS Responses to Queries |

The CDR-Lite Grails solution implements a Spring Model View Controller (Spring MVC) web application framework. Spring MVC is an extensible MVC making it perfect for Grails. The Grails servlet extends Spring's [DispatcherServlet](http://www.springframework.org/docs/api/org/springframework/web/servlet/DispatcherServlet.html) to bootstrap the Grails environment, there is then a single Spring MVC controller called [org.codehaus.groovy.grails.web.servlet.mvc.SimpleGrailsController](http://svn.grails.codehaus.org/browse/grails/trunk/grails/src/web/org/codehaus/groovy/grails/web/servlet/mvc/SimpleGrailsController.java?r=HEAD) that handles all Grails controller requests.

The SimpleGrailsController delegates to a class called [org.codehaus.groovy.grails.web.servlet.mvc.SimpleGrailsControllerHelper](http://svn.grails.codehaus.org/browse/grails/trunk/grails/src/web/org/codehaus/groovy/grails/web/servlet/mvc/SimpleGrailsControllerHelper.java?r=HEAD) 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:

1. Parses the URI into its components (controller name, action name, id etc.)
2. Looks up a GrailsControllerClass instance for the URI (see Table 2)
3. Creates a new Grails controller instance
4. Configures the controller instance's dynamic methods and properties
5. Retrieves a scaffolder if scaffolding is enabled for the controller
6. Gets a reference to the closure action to execute for the URI
7. Increments flash scope moving it on to its next state
8. Gets the view name for the URI
9. Executes any interceptors that have been registered (eg: Spring Security)
10. Executes the closure that is the controller action if the “before” interceptor didn't return false
11. Creates a Spring MVC ModelAndView instance from the view name and the model returned by the closure action
12. Executes any “after” interceptors registered, passing the returned model to the interceptor
13. Returns the Spring MVC ModelAndView instance



Figure 4- Grails framework in context of Spring MVC

CDR-Lite Grails is implemented as a standard Grails 2.4.4 project. CDR-Lite forms are implemented as GSPs; Data records and static members are Grails Domain Classes (see Table 1); XML interfaces are defined as Grails Services. Beginning with Figure 5, the directory structure is described, showing how the elements of CDR-Lite map into Figure 4.

The configuration file, shown in Table 2, controls which roles have access to which controllers. The restrictions imposed by this configuration file limit data access by both roles and controllers.

Table 2 - Spring Security Configuration File Config.groovy, Showing Roles and Restrictions

|  |
| --- |
| 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 everyting 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 5 shows a high-level view of the directory structure. This is, basically, an extension of the default Grails directory structure.

The conf, controllers, domain, jobs, and services directories are described in the following sections.

The cdrlite/web-app directory is the location of the standard directory structure defined in the J2EE specification, such as WEB-INF and META-INF.

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

Figure 5 - High Level Overall View of CDR-Lite Directory Structure

The conf directory, shown in Figure 6, stores various files used in configuring CDR-Lite, and the data source (DataSource.groovy). When needed, other configuration files go in this directory. Grails removes the need to add configuration in XML files for the CDR-Lite. 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 (for example LoginController) and is in the Controllers folder is a controller. The controllers related to base classes are stored under the Controller directory, shown in Figure 7.

|  |
| --- |
| +---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 6 - CDR-Lite conf Directory Details

Figure 7 shows the controllers directory that 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-Lite 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.” Additionally, Table 3 shows controllers unrelated to Domain Classes.

Table 3 - Controllers Not Associated with Domain Classes

| **Controllers Not Associated with A Domain Class** | **Description** |
| --- | --- |
| backoffice | Supports monitoring of system functioning, activities and RESTful services |
| Errors | Supports the “Page not found” screen, when requested information is not available. |
| Login | Supports the authentication of a user when they attempt access to the web site |
| Help | Supports the system help screens that are available to users |
| helpFileUpload | Supports the upload of user manuals, SOPs etc. for the users |
| Home | Supports the initial page that a user sees when they login. This home page is a function of the individual user, and their role. |
| userLogin | Supports the reporting of users, and the information stored about those users, such as affiliation |
| Logout | Destroys the session |

|  |
| --- |
| \---cdrlite  +---grails-app  | +---controllers  | | \---nci  | | \---bbrb  | | +---cdr  | | | +---datarecords  | | | +---forms  | | | | \---blood  | | | +---prc  | | | +---staticmembers  | | | \---util  | | \---cdrlite |

Figure 7 - CDR-LITE controllers Directory Structure

The grails-app/domain directory, shown in Figure 8, stores the domain classes. A description of the individual domain classes is in Table 1. Each of the files is a .groovy file, with the file name the name of the domain class. The directory structure divides the files into core functionality (e.g., data records, prc, and util) and forms. GORM manages all domain classes found in the domain. Methods are dynamically added to aid in persisting the class's instances. The files in this figure map into the Grails Domain box in Figure 4.

|  |
| --- |
| \---cdrlite  +---grails-app  | +---domain  | | \---nci  | | \---bbrb  | | \---cdr  | | +---authservice  | | +---datarecords  | | +---forms  | | | \---blood  | | +---prc  | | +---staticmembers  | | \---util |

Figure 8 - CDR-Lite domain Directory Structure

Figure 9 shows a collection of directories defined in the Grails scaffolding.

The grails-app/ i18n directory is designated to hold internationalization information for the web site. This directory contains the file messages.properties, which contains settings for error messages used in data validation. Only English messages are available.

The grails-app/jobs/cdrlite directory contains groovy scripts that can be run on a periodic basis (“cron” jobs). These jobs are coordinated via the Quartz package.

Table 4 - CDR-Lite Quartz Scheduler Jobs

| **CDR-Lite Quartz Jobs** | **Description** |
| --- | --- |
| TextIndexJob.groovy | Updates indexing of text fields for Lucene searching |

The grails-app/services/nci/bbrb/cdr directory is the repository for the CDR-Lite Grails service classes. A service class is a Plain Old Groovy Object (POGO) frequently with the name starting with a domain class, and ending with “Service.groovy.” These service classes are Spring-loaded groovy beans. The following table (Table 5) describes the CDR-LITE services that are not associated with a Domain Class.

Table 5 - Service Classes Not Associated with a Single Domain Class

| **Non Domain Class Service** | **Description** |
| --- | --- |
| AccessPrivilegeService | Service supporting Spring in checking the access privileges of the user and role |
| ActivityEventService | Implements user e-mail notification when various events take place (such as a case collected) |
| BloodService | Services supporting the Blood form |
| CandidateService | Services to support creation of Candidates |
| CaseStatusService | Service to support setting and changing of Case Status |
| HealthHistoryService | Supports saving the Health History Form |
| HubIdGenService | Service to generate unique identifiers |
| LocalPathReviewService | Service to support saving the Local Pathology Review form |
| PrcReportService | Service supporting the PRC report form |
| ProcessingService | Service to support shipping and processing events, as XML payloads, when these are received from Van Andel via REST over HTTPS |
| QueryService | Service supporting the query tracker |
| SendMailService | Service supporting mail notification of users when triggers occur |
| SlideSectionService | Service supporting the Slide Sectioning form |
| TextSearchService | Service supporting the Lucene searches of records |
| TissueGrossEvaluationService | Services supporting the tissue Gross Evaluation form |

The grails-app/taglib/cdrlite directory, shown in Table 6, contains custom Groovy code that dynamically generates the HTML associated with GSP tags in the forms. <g:{tagname} {attributes} /> is the form these tags take in the GSP pages.

Table 6 – Custom Groovy Tags

| **Classes for Implementing Dynamic Customized Code** | **Description** |
| --- | --- |
| CaseRecordLinkTagLib.groovy | Implements customized GSP tags for the Case Record display with a link to the CaseRecordController by id. |
| JqueryDatePickerTagLib.groovy | Implements a customized date-picker |
| MedicationAdminTagLib.groovy | Implements customized GSP tags for the BPV Medication display |
| QueryTagLib.groovy | Implements customized GSP tags for the query tracker |
| RadioButtonTagLib.groovy | Implements customized GSP tags for single-select buttons |

The grails-app/view, Table 7, has one sub-directory for each domain class. Each entry under that directory is one or more Grails Server Page (.gsp) files. Each of these files has a name describing a method in the corresponding controller class. Here are the typical methods that are auto generated whenever you create a domain class (you can create other views and controller methods – the names *should*, by convention, match)

Table 7 - Typical/Default Grails Server Pages

| Grails Server Page | Description |
| --- | --- |
| create.gsp | Produce a user interface to enter all values for a domain class, and create a new instance |
| edit.gsp | Produce a user interface 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 | Produce a list of persistent domain class objects |
| show.gsp | Produce a user interface showing the contents of an instance of the domain class |
| \_form.gsp | An include file with all the domain class attributes, it is used by the create.gsp and edit.gsp files |

|  |
| --- |
| \---cdrlite  +---grails-app  +---i18n  +---jobs  | \---cdrlite  +---services  | \---nci  | \---bbrb  | \---cdr  +---taglib  | \---cdrlite  +---utils  \---views |

Figure 9 - CDR-Lite General Purpose Directories

Figure 10 shows the cdrlite/src directory containing groovy, java, and template code.

The cdrlite/src/groovy/nci/bbrb/cdr directory contains one file, CDRBaseClass.groovy, which is the base class for all CDR-Lite Groovy classes. The CDR-Lite developers use it to create an abstract class that the domain classes have all inherited. By extending the CDRBaseClass when implementing a domain class, the domain class inherits 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 cdrlite/src/groovy/nci/bbrb/cdr/context directory contains one file, CDRApplicationEvent.groovy, which responds to all Spring class events. This is an interceptor class, which fires automatically whenever a Spring security event (such as login or update to a domain class) is triggered.

The cdrlite/src/groovy/nci/bbrb/cdr/datarecords directory contains one commonly used file, DataRecordBaseClass.groovy, which is the base class for all Domain Classes. DataRecordBaseClass extends CDRBaseClass, however it implements no new attributes.

The cdrlite/src/groovy/nci/obbr/cahub/staticmember directory contains one file, StaticMemberBaseClass.groovy, which serves as the base class for all CDR-Lite static classes. Inheriting the StaticMemberBaseClass gives every Static Member a Name and a Code attribute, and the auditable property.

The cdrlite/src/groovy/nci/obbr/cahub/utility directory contains files, for recording user access, and event logging.

|  |
| --- |
| \---cdrlite  +---grails-app  +---src  | +---groovy  | | \---nci  | | \---bbrb  | | \---cdr  | | +---context  | | +---datarecords  | | +---forms  | | +---security  | | +---staticmembers  | | \---util  | | +---appaccess  | | \---pogo  | +---java  | \---templates |

Figure 10 - CDR-Lite src Directory Structure

The web-app directory is shown in Figure 11. This is a standard web-app directory structure from J2EE. The standard js directory includes openseadragon (Open Sea Dragon GitHub Repository)[[5]](#endnote-2), used in visualizing whole slide images of specimens. Openseadragon[[6]](#footnote-6) was released under the new BSD license. Openseadragon was used for the CDR, however it is not used in CDR-Lite. The rest of the directories under web-app contain custom java script written for the CDR-Lite application, in most cases specific to each form. The WEB-INF and META-INF directories are also located here, but not shown by default in the NetBeans IDE project view. applicationContext.xml is generated and managed by grails, not the programmer.

|  |
| --- |
| \---cdrlite  +---grails-app  \---web-app  +---css  +---images  +---js  +---countdown  +---ext  +---openseadragon-bin-1.1.1  \---timeentry |

Figure 11 - CDR-Lite web-app directory Structure

Grails uses the concept of "convention over configuration." This means that typically the name and location of files is used instead of explicit configuration in XML files, hence you need to familiarize yourself with the directory structure provided by Grails 2.x

cdrlite is the main application directory, which contains the following 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, specimens,
  + i18n - Support for internationalization (i18n).
  + Services – Holds the server classes, which are Spring-managed beans.
  + taglib – contains Groovy Server Pages (GSP) custom tag libraries
  + utils – Holds a variety of codec classes[[7]](#footnote-7).
  + Views - Groovy Server Pages - The V (view) in MVC.
  + Scripts – Groovy scripts

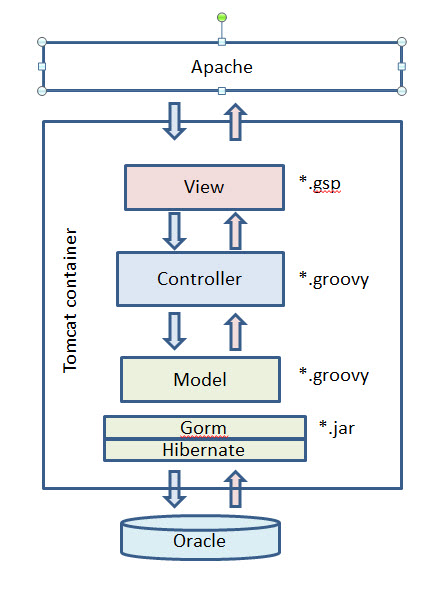
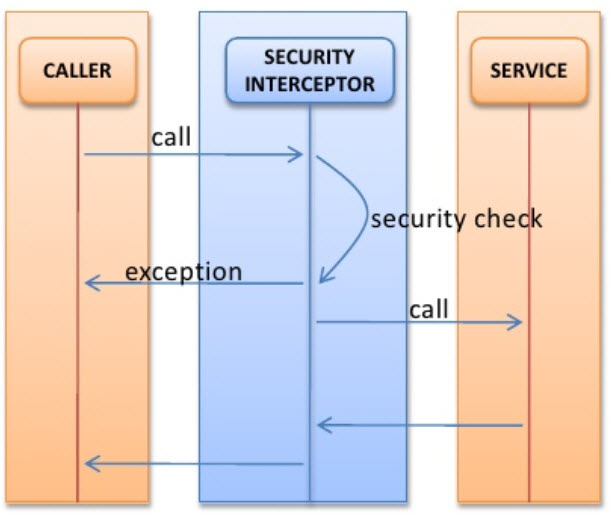


Figure 12 - Grails Application in MVC Context and Server Aspect

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

## Security Architecture

As shown in 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. Spring Security decides, for every request, whether the requesting user is (a) authenticated, and (b) authorized, before each request is fulfilled.



**Figure 13- CDR-Lite Spring Security Model**

The configuration of the Spring Security Model is shown in Table 2.

## Communication Architecture

Communications with other systems outside of the CDR-Lite happens in the following ways:

* Web services
* Email notifications
* Database CRUD operations

Each of these communications mechanisms is discussed in the following sub-paragraphs.

### Web Services

The CDR-Lite invokes and accepts external communications by RESTful web services. This is an industry standard built on top of the HTTP protocol, using HTTPs GETS and POSTS. Data exchanged via the RESTful web services are XML documents.

Table 8 - RESTful Services Available with CDR-Lite

| **RESTful Service Type** | **Purpose** |
| --- | --- |
| Processing Event | Processing event triggers an e-mail to a configurable distribution list. |

### Email notifications

The CDR-Lite sends email notifications using the industry standard SMTP protocol. When various triggers are detected by the CDR, email notifications are sent to pre-defined email groups. Those members of that individual group receive notifications at the same time. The text of the message describes what triggered the email; giving specifics (not containing PII or PHI) so that the recipient can take the appropriate action. Recipients are defined by Exchange distribution lists, and in the application settings, which can be modified by system administrators.

### Database CRUD operations

The CDR-Lite uses JDBC to execute Create, Read, Update, and Delete (CRUD) operations on a dedicated, local instance of PostgreSQL. By default, and in DataSource.groovy, the CDR-Lite expects to find PostgreSQL running on the same server, localhost:5432. 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 Lite database, nor does CDR-Lite access any other database instances.

# System Design

## Database Design

A high-level overview of the database design is shown in Figure 14

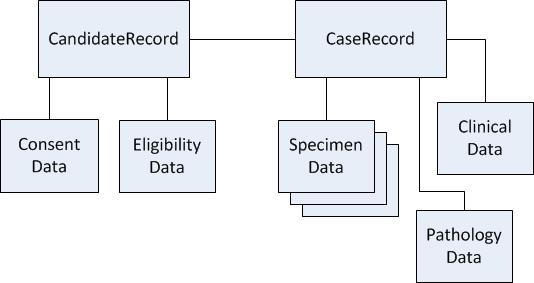


Figure 14- High-Level Overview of CDR Lite Tables and relations

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

For those candidates who are eligible, and a consent has been acquired, then a Case Record (domain class: caseRecord) is created. This 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, the specimens are reviewed by a pathologist 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 also very study specific, so this class is often modified, reflecting study-specific 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 9 shows the mapping between tables in the database and Domain Classes.

Table 9 - Mapping between Database Tables and Domain Classes in CDR

| **Database Table** | **Domain Class** |
| --- | --- |
| ACTIVITY\_EVENT | activityEvent |
| AUDIT\_LOG | Audit Log |
| blood\_aliquot | Aliquots for the Blood form |
| blood\_collection\_tube | collection tubes for the blood form |
| blood\_draw | draws for the blood form |
| cdr\_role | Spring Security: role |
| cdr\_user | Spring Security: user |
| cdr\_user\_role | Spring Security: user to role many-to-many mapping table |
| DEVIATION | deviation |
| DR\_CANDIDATE | candidateRecord |
| DR\_CASE | caseRecord |
| DR\_IMAGE | ImageRecord |
| Dr\_photo | PhotoRecord |
| DR\_PROCESSEVT | processingEvent |
| DR\_SLIDE | SlideRecord |
| DR\_SLIDE\_PROCESS\_EVENTS | Slide processing events |
| DR\_SPECIMEN | specimenRecord |
| DR\_SPECIMEN\_PROCESS\_EVENTS | Specimen processing events |
| FORM\_BLOOD | Blood form |
| FILE\_UPLOAD | fileUpload |
| QUERY | Query |
| QUERY\_ATTACHMENT | Query attachment |
| QUERY\_RESPONSE | Query Response |
| form\_cancer\_history | CancerHistory |
| form\_clinical\_data\_entry | ClinicalDataEntry |
| form\_consent\_verification | ConsentVerification |
| form\_demographics | Demographics |
| form\_general\_medical\_history | GeneralMedicalHistory |
| form\_health\_history | HealthHistory |
| form\_med\_history | MedicationHistory |
| form\_screening\_enrollment | ScreeningEnrollment |
| form\_slide\_prep | SlidePrep |
| form\_slide\_prep\_dr\_slide | Generated by GORM |
| form\_slide\_section | SlideSection |
| form\_slide\_section\_dr\_specimen | Generated by GORM |
| form\_social\_history | SocialHistory |
| form\_surgery\_anesthesia | SurgeryAnesthesia |
| form\_therapy | Therapy |
| form\_tissue\_gross\_evaluation | TissueGrossEvaluation |
| form\_tissue\_process\_embed | TissueProcessEmbed |
| form\_tissue\_receipt\_dissect | TissueReceiptDissection |

Table 1 shows the tables in the CDR-Lite for basic information handling. When new forms are entered into the CDR-Lite, additional tables and fields are automatically added by the software, through the Hibernate mechanism in CDR-Lite. This flexibility allows the data entered into CDR-Lite to dynamically change, meeting developing requirements. Hibernate will not delete columns in the database however. If an attribute is deleted in a domain class, that must be done manually at the database level.

In the case of many-to-many relationships, Grails/GORM define an intermediate table to hold the foreign keys of the two domain classes participating in the many-to-many relationship. In the table above, the slide prep form and the dr\_slide domain class, have a many-to-many relationship, and the form\_slide\_prep\_dr\_slide table is created as a result.

## Data Conversion and DE-Identification

No data conversion is done by the CDR-Lite. Users enter data in one or more electronic forms, and data get 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 because of the value entered in other fields.

PHI data may be entered on several of the forms managed by CDR-Lite. The level of PHI stored is restricted to a limited data set (LDS). This LDS data is stored in the underlying database. Access to areas of the CDR-Lite containing LDS data is controlled both by user entitlements and role, and by validation against Spring Security. If authorized, full LDS data may be displayed on screens and reported; otherwise, data is de-identified through dynamic content redaction. Examples of de-identified elements include birth dates, dates of procedures, or dates in relation to which procedures were performed or may be deduced, etc. This dynamic redaction is performed in the GSP page. If a field may contain PHI, it must be coded as such using the custom jQueryDatePicker tag library. See below:

<g:jqDatePicker LDSOverlay="${bodyclass ?: ''}" name="surgeryDate" value="${surgeryAnesthesiaInstance?.surgeryDate}"/>

## User Interface Design

The CDR-Lite’s User Interface (UI) Design is based on standard web templating, using SiteMesh, scripting and cascading style sheets (CSS). HTML, JavaScript and CSS are embedded in Groovy Server Pages, or included from the WEB-INF/js and WEB-INF/css folders. The rendered output is a standards compliant, cross-browser compatible HTML page. All pages in CDR-Lite include the cahubTemplate. Changes to the overall look and feel, banners, footers, etc., are handled in the cahubTemplate, which can be found under Views and Layouts in the layouts folder.

The UI requirements for the CDR-Lite were minimal with respect to how the site was to look to the BSS users. The user interface is designed to look and act much 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 designed to capture clinical data were based on the BPV SOPs for data capture. All users share the same style of interface, but fields, and entire pages, are restricted from some users.

### Users, Roles and Audiences

The CDR had to support different user types, roles and privileges. Some of these users were external to the CDR, some internal. These users either had read-write access to the data, or just read-only. Some roles could only see certain aspects of the programs supported by the CDR, while others could only see data generated by their particular organization. Figure 19 is a table showing the roles and privileges supported by the CDR-Lite. Each role is configured and validated against Spring Security upon login. A user’s privileges are tied with their role, and organization. This combination determines their privilege and access level.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Role** | **Write** | **LDS (access to PII)** | **Global Access** | **Notes** |
| **BSS** | Y | Y | N | Biospecimen Source Site staff role |
| **DM** | Y | Y | Y | DM = Data Manager |
| **PRC** | Y | N | Y | PRC = Pathology Resource Center Pathologist |
| **LDS** | N | Y | Y | LDS = Read only with access to HIPPA 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 accounts to Web Service APIs |

Figure 15- CDR-Lite’s User Role and privilege table

Access to study-specific and functional areas of the CDR-Lite is available through a user’s homepage, as shown in Figure 16. Depending on a user’s entitlements, they may see a different homepage or be restricted to certain areas. Some power user roles, such as DM, LDS and Super have the ability for raising and lowering their privileges as needed.

Figure 16- Example CDR-Lite Home Page

Figure 17- CDR Home Screen for BBRB and Leidos Users

### Triggers

The CDR-Lite uses a mechanism, called “Triggers”, which react when some predefined event happens, or a business rule is applied, by sending email messages to the appropriate users. Each trigger contains a number of elements, including specific 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 by email when the event happens. The communications mechanism for emails is discussed in section 4.6.2. Individual triggers may be customized to include additional information in the body of the message (such as case ID, or individual field values not containing PII/PHI).

The full list of triggers and a description of what event is associated with that trigger is shown in Figure 18. When a trigger fires, it creates an SMTP message specific to the event. That SMTP message hands off to an external mail server, which delivers the message to the appropriate experts. Which individual users get the message is dependent on the SMTP mail list description; by adding a person to a given distribution list, that person automatically gets all future messages. The users then take the appropriate action.

| **Name of Trigger Distribution** | **Triggering Event** |
| --- | --- |
| APERIO\_IMAGE\_DISTRO | Notification: Whole slide images are available at the CBR for a given study. |
| NEW\_QUERY\_TRACKER\_DISTRO | Notification: A new Query was created by Data Management. |
| CDRLITE\_ADMIN\_DISTRO | Notification when a case is created or when case status changes |

Figure 18- Mail Distribution List for Various Triggers

###### Key Terms

The following table provides definitions and explanations for terms and acronyms relevant to the content presented within this document.

| **Term** | **Definition** |
| --- | --- |
| ABCC | Advanced Biomedical Computer Center |
| AOP | Aspect oriented programming is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns. |
| BPV | Biospecimen Preanalytic Variables – a study sponsored by BBRB, which used CDR for managing study specific data. |
| BSS | Biospecimen Source Site – Institute from which human tissue is initially collected. |
| CDR | Comprehensive Data Resource |
| DM | Data Management – the people and activities intent on preserving data integrity. |
| 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 | NIH Common Fund's Genotype-Tissue Expression program |
| HHS | U.S. Department of Health and Human Services |
| LDS | Limited Data Set – a reflection of the central data where PHI/PII data has been protected. |
| LIMS | Laboratory Information Management System |
| PII | Personally Identifiable Information - Individually identifiable health information |
| PHI | Protected Health Information - Health information, including demographic information; Relates to an individual’s physical or mental health or the provision of or payment for health care |
| RESTful | A type of Internet service interface typically between programs, which exchanges information. |
| SOP | Standard Operating Procedure – a detailed document describing precisely the performance of a protocol. |
| UI | User Interface – typically refers to the web-based graphical user interface which enables the various groups to enter/retrieve data from the CDR-DS or CDR-AR. |
| XML | Extensible Markup Language. A markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all free open standards. |

###### CDR-Lite Design Approval

The undersigned acknowledge that they have reviewed the CDR-Lite Design Document, and agree with the information presented within this document. Changes to this CDR-Lite Architecture and Design 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: |  |  |  |

1. <http://biospecimens.cancer.gov/researchnetwork/lifecycle.asp> [↑](#footnote-ref-2)
2. <https://commonfund.nih.gov/GTEx/index> [↑](#footnote-ref-3)
3. <http://biospecimens.cancer.gov/programs/bpv/default.asp> [↑](#footnote-ref-4)
4. http://www.hopkinsmedicine.org/institutional\_review\_board/hipaa\_research/limited\_data\_set.html [↑](#footnote-ref-5)
5. [↑](#endnote-ref-2)
6. https://openseadragon.github.io [↑](#footnote-ref-6)
7. See [http://grails.org/doc/2.3.1/guide/single.html#codecs.](http://grails.org/doc/2.3.1/guide/single.html%23codecs.) [↑](#footnote-ref-7)