

Contents

- 1 Introduction
 - ◆ 1.1 Background
 - ◆ 1.2 Scope
 - ◆ 1.3 Document Change History
 - ◆ 1.4 Project Team
 - ◆ 1.5 Contacts and Support
 - ◆ 1.6 Additional Documentation
- 2 Overview
 - ◆ 2.1 Domain Driven Design
 - ◆ 2.2 Layers and Tiers
 - ◆ 2.3 SOA
- 3 Security
 - ◆ 3.1 TLS
 - ◆ 3.2 Dorian
 - ◆ 3.3 Grid Authentication
 - ◆ 3.4 Authorization
 - ◆ 3.5 Site Level Security
- 4 Multi-site
 - ◆ 4.1 Approach
 - ◆ 4.2 Services
 - ◆ 4.3 Configuration
 - ◆ 4.4 Reliability
- 5 CCTS Integration
 - ◆ 5.1 Services
 - ◆ 5.2 Security
 - ◆ 5.3 Hotlinks
- 6 Notifications
 - ◆ 6.1 Approach
 - ◆ 6.2 Hibernate Interceptors

- ◆ [6.3 Rules Engine](#)
- ◆ [6.4 Scheduler](#)
- ◆ [6.5 JavaMail](#)
- ◆ [6.6 Freemarker](#)
- [7 Deployment](#)
 - ◆ [7.1 Stand-alone](#)
 - ◆ [7.2 Multi-site/Hosted](#)
 - ◆ [7.3 CCTS](#)
- [8 Appendix A: Glossary](#)

Introduction

Background

The Cancer Center Participant Registry (C3PR) is a web-based application used for end-to-end registration of patients to clinical trials. This includes capturing the consent signed date, eligibility criteria, stratification, randomization, and screening. Clinical workflows are enabled by both subject- and study-centric views into the registration process. C3PR can be run in a standalone mode where study definitions, investigators, study personnel, and sites are entered into the system, or C3PR can be run in an integrated mode with the caBIG Clinical Trials Suite (CCTS). C3PR also enables multi-site clinical trials where registration information is entered locally at affiliate sites and the registration is completed by call-out to the coordinating site.

Throughout the development of C3PR, a number of elaborator and adopter sites are actively being engaged to help define requirements and test the application. Our primary elaborators include Duke, Wake Forest, Mayo, Westat, CALGB, CCR, and the Coalition of Cooperative Groups. Our primary adopters include Duke and Wake Forest with engagement of Georgetown and CCR.

C3PR release 1 was developed by Nortel Solutions and released in 2006. Release 2 was developed by Duke Cancer Center in collaboration with SemanticBits, LLC and was released in March, 2008. We are currently in the next phase of development with releases slated for the end of September, 2008 and March, 2009.

Scope

This document is focused on the technical design of C3PR. The primary audience is system architects and software developers or anyone else with a technical background. The document is scoped to high level architectural considerations but also delves into areas that the reader may find useful for implementing C3PR or a similar system.

Document Change History

Version Number	Date	Contributor	Description
v1.0	12/19/2006	Patrick McConnell	Created template
v1.1	2/28/2007	Manav Kher	Added details
v1.2	05/04/2007	Manav Kher	Added section on CSM
v1.3	06/08/2007	Manav Kher	Incorporated feedback from colleagues
v.1.4	10/02/2007	Manav Kher	Site level security
v.2.5	9/12/2008	Patrick McConnell, Kruttik Agarwal, Ram Chilukuri	Rewrite for 2.5 release

Project Team

C3PR is a highly collaborative project made up of domain and technical experts from Duke Cancer Center, SemanticBits, Wake Forest Cancer Center, Mayo Clinic, CALGB, and Westat.

Team	Members	Team	Members
Duke	<ul style="list-style-type: none"> • Jamie Cuticchia (PI) • Bob Annecharico (Project Director, Co-investigator) • Pankaj Agarwal (Project Manager) • Mohammad Farid (DBA) • Peter Le (IT Analyst) • Vijaya Chadaram, RN (Subject Matter Expert) • Emily Allred (Admin) 	SemanticBits	<ul style="list-style-type: none"> • <u>Ram Chilukuri (Technical Director, Architect)</u> • <u>Patrick McConnell (Architect)</u> • Kruttik Aggarwal (Lead Developer) • Ramakrishna Gundala (Developer) • Vinay Gangoli (Developer) • Himanshu Gupta (Developer) • Vinay Kumar (Business Analyst) • Shilpa Alluru (QA Tester) • Elizabeth Nobriga (Technical Writer)
Wake Forest	<ul style="list-style-type: none"> • Bob Morrell (Institutional Lead, Subject Matter Expert) • Don Babcock (Technical Expert) • Lisa Dixon (Subject Matter Expert) • Claire Kimbrough (Subject Matter Expert) 	Mayo Clinic	<ul style="list-style-type: none"> • Sharon Elcombe (Institutional Lead, Subject Matter Expert)
CALGB	<ul style="list-style-type: none"> • Kimberly Johnson (Institutional Lead, Subject Matter Expert) 	Westat	<ul style="list-style-type: none"> • Steve Riorden (Institutional Lead, Subject Matter Expert)

• Amish Shah (IT Analyst)

Contacts and Support

Name	Contact
End User Forum	https://cabig-kc.nci.nih.gov/CTMS/forums/viewforum.php?f=17
Technical Forum	https://cabig-kc.nci.nih.gov/CTMS/forums/viewforum.php?f=15

Additional Documentation

End User	Analysis	Technical	Management
C3PR Main Project Page	Use Cases	Architecture Guide	Project Plan
End User Guide	Requirements Specification	Domain Analysis Model	Scrum Artifacts
Training Library	Activity Diagrams	Implementation Model	Adoption Plan
Installation Guide	BRIDG Compliance Report	Multisite Deployment Guidelines	Communications Plan
Configuration Guide	BAM Compliance Report	Deployment Diagrams	Test Plan
Release Notes	COPPA Gap Analysis		Test Logs
Tear Sheet			Arc Requests
			Lessons Learned
			C3PR caBIG License

Overview

Domain Driven Design

The C3PR project is developed using the principles of Domain Driven Design (DDD). This approach puts the domain at the forefront of the development effort by focusing on the domain model and domain logic. A key aspect of this approach is that the domain model forms a common language (called a ubiquitous language) for describing and understanding requirements. This language is used throughout the model, code, use cases, requirements, etc. This facilitates clear communication between team members and especially with stakeholders. For example, we use the term StudySubject throughout the project artifacts to refer to a subject that is assigned on a study.

While Model-driven Architecture (MDA) can be a complementary approach to DDD, its goals are somewhat different. MDA is concerned primarily with deriving the code that makes up a system directly from a model, whereas DDD is concerned more with defining better, more correct domain models that are used to develop the system. In C3PR, we derive our data objects (known as entities and value objects in DDD) and database structure from the model, though not in an automated way. Instead, we use the domain model to inform the creation of these classes using the notion of a ubiquitous language.

Layers and Tiers

DDD describes a set of programming artifacts that are used for implementation of a domain:

- **Entities:** an object in the domain model that is not defined by its attributes, but rather by a thread of continuity and identity.
- **Value Objects:** an object that has no conceptual identity. These objects describe a characteristic of a thing.
- **Repository:** methods for retrieving domain objects should delegate to a specialized 'repository' object such that alternative implementations may be easily interchanged.
- **Factory:** methods for creating domain objects should delegate to a specialized 'factory' object such that alternative implementations may be easily interchanged.
- **Service:** when an operation does not conceptually belong to any object, it is implemented in services.

We leverage these concepts to implement domain models and domain logic. However, we also leverage an N-tiered architecture:

- **Presentation:** Grid Services, Web UI
- **Middle Tier:** DDD artifacts such as entities (DAOs), repositories, and services
- **Database:** Postgres, Oracle, or MS SQLServer

These tiers are implemented using the following technologies (among others)

- **Grid Services:** caGrid 1.2
- **Web Interface:** Spring
- **Entities (DAOs):** Hibernate
- **Repositories:** Java classes
- **Services:** Java classes
- **Database:** Postgres, Oracle, or MS SQLServer

SOA

C3PR is based on a Service Oriented Architecture (SOA) whereby services expose domain logic for internal C3PR use (DDD Services) and external consumption (Grid Services). These services abstract domain logic into a series of operations on secure services. C3PR provides the following services, which implement C3PR multi-site use cases:

- **Registration Service:** handles multi-site registration of subjects onto trials. This service is designed to be exposed at a coordinating site to handle the registration workflow.

- **Study Service:** handles multi-site distribution of study data. This service is designed to be exposed at an affiliate site to accept study definitions from a coordinating site.

These services are exposed as caGrid 1.2 secure grid services, which provides a language agnostic service layer. Ultimately, these caGrid services will provide a presentation layer to expose DDD Services. However, at this point, domain logic is encapsulated within the grid service implementation and not exposed as common DDD Services.

Security

The requirements of the C3PR security architecture are to ensure the privacy and integrity of all network communication, and to enforce authentication and authorization before granting access to operations and data. Privacy and integrity of data in transit through the network are ensured by using Transport Level Security (TLS). Access to data at rest in a database is protected using basic (username/password) authentication mechanisms. Authentication is enforced using a combination of Public Key Infrastructure (PKI) and basic authentication. Authorization policy is constructed and maintained using NCICB's Common Security Module (CSM). Authorization policy is enforced using custom components in the web application layer and the caGrid Authz module, which enables local and external (grid-wide) policy to be enforced in parallel.

PKI-based authentication works as follows. Users and services are provisioned with X509v3 certificates that have been signed by widely-trusted certificate authorities. During negotiation of security communication between parties, party A authenticates party B by verifying that the certificate of party B has been signed by a certificate authority that party A trusts. Both uni- and bi-directional (mutual) authentication can be required. Basic authentication involves looking up the requesting parties credentials (username and password) in some credential provider. Grid users have a proxy certificate (short-lived X509 certificate) and are authenticated using PKI authentication.

C3PR can operate in one of three primary modes:

- **Stand-alone:** users are authenticated and authorized using standard CSM mechanisms
- **Multi-site/hosted:** users are authenticated using a caGrid Authentication Service that wraps the CSM identity provider (IdP) and authorized using standard CSM mechanisms. For purposes of data exchange, users are represented with X509 certificates, which are mapped to CSM users by user name.
- **CCTS:** users are authentication using the Dorian IdP and authorized using standard CSM mechanisms. For purposes of data exchange, users are represented with X509 certificates, which are delegated to middle-tier components (such as caXchange) and mapped to CSM users by user name.

The following sections describe how security issues are addressed at each layer of the C3PR security architecture.

TLS

The C3PR web application runs in a servlet container. Web browser clients make requests to the container, and code running in the container sends and receives messages through grid services. To secure communication between the web browser and the web application, the servlet container is configured to accept connections over HTTPS, and the web application is configured to require that pages containing sensitive data can only be accessed over HTTPS. In this way, all data communicated between the browser and the application are encrypted. To secure communication between the web application and grid services, the caGrid 1.2 services are configured as secure services to accept SSL connections and the client is configured to use a SSL connection to the service.

Dorian

Dorian is the Grid User Account Management system provided by caGrid 1.2. Dorian acts as a bridge between external security domains and the grid. Dorian allows users to use their existing credentials (external to the grid) to authenticate to the grid.

Dorian comes with a default identity provider. This allows users to register for an account and obtain grid credentials through Dorian. We leverage the Dorian default IdP for requesting grid credentials for users in a CCTS deployment and a standard caGrid Authentication service for users in a multi-site/hosted mode. These grid credentials can then be used to authenticate to other systems, thus providing a Single Sign On (SSO) between these systems:

- Proxy Creation Workflow
- Client authenticates with Local IdP
- Client creates public/private key pair to use for grid proxy.
- Client requests Dorian to create a grid proxy.
- Dorian verifies that the SAML assertion provided by the user is signed by a Trusted IdP and that the user has a valid account.
- Dorian locates the user's grid credentials, private key and certificate
- Dorian uses the public key provided to create a proxy certificate and signs it with the user's private key
- Dorian returns the proxy certificate to the user.
- The user may now use the proxy to authenticate to grid services

Grid Authentication

The first step in the security workflow is authentication, which is the process by which a trusted organization claims a user is who they claim to be. This is implemented using the caGrid AuthenticationService. The goal is for the user to get a grid user certificate to be used in authorization. The user submits their credentials (user name and password) to the AuthenticationService, which has an institutional IdP plugged in. This component performs the necessary steps to determine that the user credentials are valid. In the case of the Duke IdP plugin, the user name and password are validated against the Duke Domain Controller, which uses NT Security. Once validated, the AuthenticationService returns a SAML Assertion to the user, which asserts that the user is who they claim to be. The SAML Assertion can then be passed off to a Dorian Service to generate

a grid user certificate. Grid users must provide a proxy certificate which is validated using PKI. Authentication of grid users is enforced by the WEBSSO module. See the WEBSSO section for details.

Authorization

The second step in the security workflow is authorization, which is the process by which a system grants the user access to specific resources. Since C3PR can be accessed by both local and external users, authorization policy must be expressed in a way that can apply to both types of users. The general approach is to assign privileges to groups rather than users. Also, since C3PR provides a web interface and grid interfaces, the authorization policy must be expressed in a way that it can be applied to all tiers. Since entities are common to all tiers, the approach taken in C3PR is to express authorization policy strictly in terms of the domain objects and the actions that can be performed on them. So, for example, the CSM policy will never use an HTTP URL or grid service interface names as the objectId for a CSM protection element. The Role-based authorization policy applies in two ways. First is access to URLs ? roles are assigned access permissions to the pages they need to function. Second is attribute-based ? some pages mutate based on the roles granted to the logged-in user. Certain links or forms will only appear if a user has a role that grants him or her access to the target page. C3PR also uses identity-based authorization for specific resources. For example, it is possible to limit access an individual item to one or more users. Authorization is enforced through a combination of a Spring interceptor, custom JSP tags, and the Acegi framework.

The Spring DispatcherServlet passes an incoming request to the chain of interceptors before delegating to the Controller. One of these interceptors is the URLAccessCheckInterceptor, which verifies if the logged in user is authorized to access the page that he is requesting, based on his role. If the user does not have privileges to access a page then the access denied page is displayed. The URLAccessCheckInterceptor uses the CSM API to determine the roles of the logged in user.

Similarly, custom JSP tags control the rendering of a page based on the user's role, so that only sections which expose functionality and data to which the user is authorized are displayed. This works fine for local users because they can be provisioned with roles using the organization's CSM. However, it is impractical to provision all possible grid users. Instead, the Authz component is plugged into CSM to enable grid-based authorization policy to be enforced in parallel with local policy.

Site Level Security

C3PR manages patient registrations for multiple healthcare sites and cancer centers. Its very important to control access to users and only allow access to data that pertains to the healthcare site that the user belongs to. For example, a Registrar at Duke comprehensive cancer center should only be able to view Registrations and Protocols that are being conducted or managed by Duke. This is done by provisioning authorization data in CSM and controlling access to data by applying custom filters at the different layers within the c3pr application.

Multi-site

Approach

Multi-site studies are special in the sense that different sections of the life-cycle of a study and the registrations fall within different responsibilities on the different healthcare sites (Coordinating center, Funding Sponsor, Affiliate site, etc.) associated with the study depending on the role they take on the study. To efficiently manage multi-site studies, it becomes imperative that a suitable SOA is designed.

Services

C3PR can be configured to operate in a multi-site environment by setting the 'Enable multisite switch' to 'true'. The following grid services are exposed by C3PR:

- **StudyService** : This service exposes all the study life-cycle specific APIs such as 'createStudy(Study)', 'openStudy(StudyIdentifiers[])', 'activateStudySite(StudyIdentifiers[], StudySiteNCIIntstituteCode)', etc.
- **RegistrationService** : This service exposes all the study life-cycle specific APIs such as 'createRegistration(StudySubject)', 'randomize(StudySubjectIdentifiers[])', 'transferSubject(StudySubjectIdentifiers[], ScheduledEpoch)' etc.

Though C3PR is designed to support any messaging protocol like web services, JMS, grid services etc, it only ships with grid service capability.

Configuration

The multi-site messages will flow between a affiliate study site and the coordinating center and vice versa but never will two study sites talk to each other. Keeping this in mind, we can conclude that both the coordinating center's C3PR instance and affiliate site's C3PR instance need to be configured in multisite mode and need to know about each others' services.

To send a multi-site message from one C3PR system to another, the following needs to be known:

- Service URL (e.g. <https://<some.study.organization.url>/wsrf/services/cagrid/StudyService>)
- Service Name (e.g. StudyService)
- Service API (e.g. createStudy(Study))
- Message Payload (e.g. Study, StudySubject)

These properties uniquely identify a study organization and the correct API to be called on the correct service with the correct payload. C3PR defines it as an 'endpoint'. It leverages sound OOAD concepts to design the related classes and uses 'factory design pattern' to create endpoints at runtime. By using java reflection to invoke these endpoints, C3PR efficiently tackles the different flavors of endpoints (GridEndPoint, JMSEndPoint etc).

Reliability

It is evident that such an environment is subject to possible disruptions (network clogging, system exception, database exceptions, and most importantly business exceptions). It becomes a necessity to provide a robust exception handling mechanism to track failures. C3PR incorporates proper contingency planning for these scenarios and provides a capability of resending messages if they failed because of a business exception.

CCTS Integration

C3PR integrates with the caBIG Clinical Trials Suite (CCTS), an enterprise clinical trials system being designed primarily for use in trial sites. The suite is comprised of a collection of interoperable modules covering a broad range of key areas in cancer clinical trials management. These include patient registration via C3PR, patient scheduling via PSC, adverse events reporting via caAERS, lab analysis via LabViewer, and clinical data management via C3D. Integration between these applications is centered around four key scenarios: Study Creation, Register Subject, Load Labs in CDMS, and AE-Triggered Schedule Change. The implementation is based upon the caGrid infrastructure with caXchange as the Enterprise Service Bus for reliable message routing and GAARDS providing robust security.

Services

C3PR is the primary provider of study and registration data. The C3PR team has developed stubbed service skeletons for study and registration data that are reused and implemented by each of the other project teams. Once a study has been created in C3PR and opened for accrual, the user can send the study to caXchange to be routed. C3PR will display the results of the messaging. The caXchange endpoint can be configured in the C3PR configuration page. Once a registration is created, it is automatically routed to caXchange in a CCTS deployment. If the return message is successful and includes a patient position StudySubject identifier, then it is recorded in the database.

Security

As mentioned in the security section of this document, C3PR works with the grid security infrastructure. WebSSO is integrated with C3PR through an Acegi connector. Users that are not logged in are automatically redirected to the WebSSO page. The user logs in, credentials are delegated, and the user is redirected back to C3PR. The Acegi connector then retrieves the user proxy from Dorian through the delegation ticket from the Credential Delegation Service. Messages are sent to the caXchange grid service using the user's grid credential.

Hotlinks

C3PR provides the ability to hot-link (sometimes known as deep-linking) to LabViewer, PSC, and caAERS. The StudySubject identifier is passed in the URL to these applications, allowing the user to be directed directly to a subject registered to a study. Furthermore, the window title for the URL can be specified. There are three main modes: `_blank`, `_self`, and `app` where `app` is the name of the app, such as `caers`. This allows for organizations to customize how windows are handled when linking between applications.

Notifications

Approach

Notifications are set up by the administrator for notifying Personnel of specific events. Notifications are sent by Email and the list of recipients can be configured as desired. The administrator can select from a list of Events and Reports and select an appropriate frequency for the same.

Hibernate Interceptors

The primary purpose of hibernate interceptors is to intercept strategic database calls. The idea is to identify certain pre-determined events at the time of occurrence. These interceptors provide us with the old and new values of the modified variables. This not only helps us identify events but also allows us specify the notification content by divulging the old and new values of the modified attributes. These interceptors then forward the flow of control to the rules engine to determine further action.

Rules Engine

The DROOLS rules engine is invoked by the interceptor when a specific event occurs. The role of this engine is to validate the event and determine further action. If the event does not require a notification then no action is taken, otherwise the corresponding scheduler services are called. The rules engine can be leveraged in the future for building on a broader repertoire of events and reports.

Scheduler

The intention behind using the Quartz scheduler is two-fold. First, it lets us schedule jobs for pre-determined points in time, e.g. 12:00 Noon on Fridays. In this way, it fires the scheduled job at the specified time. This allows the scheduler service used by the rules engine to schedule the email sending job at a time based on the Notification frequency. Second, it lets the rules engine delegate the notification sending responsibility and

return the flow of control to the interceptor, which in turn returns to the initial action which had fired the interceptor in the first place. Finally, the scheduled job leverages the auditing framework to generate report based notifications.

JavaMail

Javamail is used to send out email notifications by the scheduled jobs. If the notification is a "report", the email content type can be HTML. Otherwise, it is plain text.

Freemarker

FreeMarker is a template engine, which is a generic tool to generate text output (anything from HTML to autogenerated source code) based on templates. It's a Java package, and its purpose here is to replace substitution variables used at the time of defining the message template with actual values at run-time. This allows the notification message body to be more meaningful.

Deployment

C3PR has three primary deployment modes:

- Stand-alone: for use at a single cancer center running local trials or capturing registration data locally
- Multi-site/hosted: for use running multi-site trials where users from different organizations can login and register subjects and/or different instances of C3PR can exchange study and registration data
- CCTS: for use in deploying C3PR with the caBIG Clinical Trials Suite whereby messages can be sent through caXchange and applications can be navigated through hotlinks

The following sections describe these deployment modes in more detail. The goal of this is not to provide a detailed set of deployment instructions, but instead to provide details on some of the architectural considerations.

Stand-alone

When C3PR is deployed in stand-alone mode, it is recommended that two nodes be provisioned at the organization:

- Database Node
 - ◆ Will contain a single database schema that has the C3PR application database tables and the CSM tables
 - ◆ Should be placed behind the firewall because it contains PHI

- Application Node
 - ◆ Will contain the web deployment of C3PR into Tomcat
 - ◆ Should be placed behind the firewall because external access is not needed (this is not strictly required)

The primary concern is that the PHI stored in the C3PR database on the Database Node be on a secure machine behind the organization's firewall.

Multi-site/Hosted

When C3PR is deployed in multi-site/hosted mode, there are two different configurations to consider. The first is at the coordinating site, where the grid security infrastructure is deployed:

- Database Node
 - ◆ Will contain a one database that has the C3PR application database tables and the CSM tables
 - ◆ Will contain another database (MySQL) that has caGrid databases
 - ◆ Should be placed behind the firewall because it contains PHI
- Application Node
 - ◆ Will contain the web deployment of C3PR into Tomcat
 - ◆ Will contain C3PR grid services deployed into Tomcat
 - ◆ Will contain SyncGTS deployed into Tomcat
 - ◆ Will contain an AuthenticationService deployed to Tomcat and authenticating to CSM
 - ◆ Should be placed outside of the firewall
- Grid Node
 - ◆ Will contain the following grid services: Dorian, GTS, SyncGTS
 - ◆ Should be placed outside of the firewall

The primary concerns at the coordinating site include locating the Database Node behind the firewall because it contains PHI, locating the Application Node and Grid Node outside the firewall because they need to be accessed remotely, deploying the security grid services that will be accessed by affiliate institutions, and instituting the appropriate trust with affiliate AuthenticationServices.

The affiliate site will be set up similarly to the coordinating site with the exception that there is no need for a grid node. An affiliate site setup is not necessary in a hosted setting (as opposed to a multi-site messaging setting where C3PR is deploying locally at the affiliate site and remotely at the coordinating site).

- Database Node
 - ◆ Will contain a one database that has the C3PR application database tables and the CSM tables
 - ◆ Should be placed behind the firewall because it contains PHI
- Application Node
 - ◆ Will contain the web deployment of C3PR into Tomcat
 - ◆ Will contain C3PR grid services deployed into Tomcat

- ◆ Will contain SyncGTS deployed into Tomcat
- ◆ Will contain an AuthenticationService deployed to Tomcat and authenticating to CSM
- ◆ Should be placed outside of the firewall

The primary concerns at the affiliate site include locating the Database Node behind the firewall because it contains PHI, locating the Application Node outside the firewall because it needs to be access remotely (the grid services need remote access), and deploying local grid services such as SyncGTS and an AuthenticationService used for authenticating to the grid.

CCTS

When CCTS is deployed in a CCTS setting, the deployment architecture is pretty much prescribed by the CCTS deployment:

- Database Node
 - ◆ Will contain a one database that has the C3PR application database tables and the CSM tables
 - ◆ Will contain another database (MySQL) that has caGrid databases
 - ◆ Should be placed behind the firewall because it contains PHI
- Application Node
 - ◆ Will contain the web deployment of C3PR into Tomcat
 - ◆ Will contain C3PR grid services deployed into Tomcat
 - ◆ Will contain SyncGTS deployed into Tomcat
 - ◆ Will contain WebSSO deploying into Tomcat
 - ◆ Should be placed inside of the firewall because there is no need for remote access
- Grid Node
 - ◆ Will contain the following grid services: Dorian, GTS, CDS, SyncGTS
 - ◆ Should be placed inside of the firewall because there is no need for remote access

The primary concerns here related to C3PR are that everything be placed within the firewall because there is no need for external access, users are managed both within the Dorian IdP and C3PR, and authentication happens through WebSSO.

Appendix A: Glossary

Term	Definition
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
BC	Binding Component
caArray	cancer Array Informatics
caBIG	cancer Biomedical Informatics Grid
caBIO	cancer Biomedical Infrastructure Objects

C3PR_Architecture_Guide

caCORE	cancer Common Ontologic Representation Environment
caDSR	cancer Data Standards Repository
C3PR	Cancer Translational Research Informatics Platform
CDE	Common Data Element
CSM	Common Security Module
CSV	Comma Delimited
DAO	Data Access Objects
DCQL	Distributed Common Query Language
DWR	Direct Web Remoting
EA	Enterprise Architect
ESB	Enterprise Service Bus
EVS	Enterprise Vocabulary Services
GAARDS	Grid authentication and authorization with Reliably Distributed Services
GUI	Graphical User Interface
HASTE	High-level Automated System Test Environment
HTTP	Hypertext Transfer Protocol
IdP	Identity Provider
JAAS	Java Authentication and Authorization Service
JAR	Java Archive
Javadoc	Tool for generating API documentation in HTML format
JDBC	Java Database Connectivity
JMS	Java Message Service
JSP	JavaServer Pages
JUnit	A simple framework to write repeatable tests
metadata	Definitional data that provides information about documentation or other data
NCI	National Cancer Institute
NCICB	National Cancer Institute Center for Bioinformatics
NMR	Normalized Message Router
ORM	Object Relational Mapping
RDBMS	Relational Database Management System
SDK	Software Development Kit
SE	Service Engine
Semantic connector	A development kit to link model elements to NCICB EVS concepts
SQL	Structured Query Language
UML	Unified Modeling Language
WSRF	Web service resource framework