CABIO 4.0

TECHNICAL GUIDE

Center for Bioinformatics

This is a U.S. Government Work

November 21, 2007

caBIG™ cancer Biomedical Informatics Grid™

Credits and Resources

caBIO Development and Management Teams		
Development	***Technical Guide***	***Program Management***
Nadine Azie[5]	Wendy Erickson-Hirons[3]	Juli Klemm[1]
Hong Dang[4]	Shaziya Muhsin[2]	Krishnakant Shanbhag[1]
Sharon Gaheen[2]	Konrad Rokicki[2]	George Komatsoulis[1]
Norval Johnson[6]	Jim Sun[2]	Denise Warzel[1]
Sriram Kalyanasundaram[6]	Lalitha Viswanath[2]	Peter Covitz[1]
Ying Long[2]		
Doug Mason[2]		
Shaziya Muhsin[2]		
Konrad Rokicki[2]		
Jim Sun[2]		
Lalitha Viswanath[2]		
Ye Wu[2]		

[1] National Cancer Institute Center for Bioinformatics (NCICB)	[2] Science Application International Corporation (SAIC)
[3] Northern Taiga Ventures Inc	[4] Alpha Gamma Technologies
[5] Ekagra Software Technologies	[6] TerpSys

Contacts and Support	
NCICB Application Support	http://ncicb.nci.nih.gov/NCICB/support Telephone: 301-451-4384 Toll free: 888-478-4423

LISTSERV Facilities Pertinent to caBIO		
LISTSERV	***URL***	***Name***
CABIO_USERS	https://list.nih.gov/archives/cabio_users.html	Users
CABIO_DEVELOPERS	https://list.nih.gov/archives/cabio_developers.html	Developers

GForge is a cross project collaboration site for NCICB caBIO developers located at http://gforge.nci.nih.gov/projects/cabiodb/.

Release Schedule

This guide has been updated for the caBIO 4.0 release. It may be updated between releases if errors or omissions are found. The current document refers to the 4.0 version of caBIO, released in December 2007 by the NCICB.

Table of Contents

Chapter 1 Using This Guide

This chapter introduces you to the caBIO 4.0 User's Guide.

Topics in this chapter include:

Purpose

The *caBIO 4.0 Technical Guide* describes the Cancer Bioinformatics Infrastructure Objects (caBIO) model, the data represented by its objects, and its application programming interfaces.

This guide describes:

- the purpose, architecture and components of caBIO
- the APIs for accessing the caBIO system including Java, Web services, and XML-HTTP
- utilities for text searching and manipulation
- the data accessible through the caBIO APIs

Release Schedule

This guide is updated for each caBIO release. It may be updated between releases if errors or omissions are found. The current document refers to the 4.0 version of caBIO, which was released in December 2007 by NCI CBIIT.

New Features in caBIO 4.0

The 4.0 release of caBIO contains many significant updates. These include:

- Generation of the API with caCORE SDK 4.0. To facilitate the upgrade of dependent systems, a Migration Guide is provided as Appendix B of this guide.
- A new text-based ("Google™-like") search tool, Freestyle Lexical Mine (FreestyleLM) is provided
- A new Home Page replaces the previous "Happy.jsp" user interface
- Support for manufacturer-provided annotations for the following microarrays:
    - Affymetrix U133 Plus 2.0 - http://www.affymetrix.com/products/arrays/specific/hgu133plus.affx

  - Affymetrix Exon 1.0 ST - http://www.affymetrix.com/products/arrays/specific/exon.affx
  - Affymetrix 500K - http://www.affymetrix.com/products/arrays/specific/500k.affx
  - Affymetrix 100K - http://www.affymetrix.com/products/arrays/specific/100k.affx
  - Agilent 44K - http://www.chem.agilent.com/scripts/pds.asp?lPage=13064
  - Agilent 244K CGH - http://www.chem.agilent.com/Scripts/PDS.asp?lPage=51324
  - Illumina 550K - http://www.illumina.com/pages.ilmn?ID=154

Audience

The primary audiences of this guide are:

- The application developer who want to learn about the architecture and use of the caBIO APIs and utilities. These sections of the guide assume familiarity with the Java programming language and/or other programming languages, database concepts, and the Internet. If you intend to use caCORE resources in software applications, it assumes that you have experience with building and using complex data systems.
- The scientist end-user who wants to understand the data provided in caBIO (Chapter 1) and/or use the Freestyle Lexical Mine search (described in Chapter 5).

Additional caBIO Documentation

The caBIO 4.0 Release Notes contain a description of the end user tool enhancements and bug fixes included in this release.

The caBIO 4.0 JavaDocs contain the current caBIO API specification

The caCORE SDK 4.0 Developer's Guide contains detailed instructions on the use of the SDK and how it helps create a caCORE-like software system.

Organization of This Guide

This brief overview explains what you will find in each section of this guide.

- Chapter 1, this chapter, introduces this guide.
- Chapter 2 provides an overview of caCORE and caBIG™
- Chapter 3 provides an overview of caBIO.
- Chapter 4 describes the architecture of caBIO
- Chapter 5 describes the various methods for accessing the data in caBIO.
- Chapter 6 describes the Cancer Bioinformatics Infrastructure Objects (caBIO)
- Appendix A includes a listing of references to provide the caBIO users with more in depth information about relevant topics.
- Appendix B provides technical users with instructions how migrate systems that have used caBIO 3.2.1 to caBIO 4.0.

Document Text Conventions

The following table (Table 1-1) shows various typefaces to differentiate between regular text and menu commands, keyboard keys, tool bar buttons, dialog box options, and text that you type. This illustrates how text conventions are represented in this manual:

Convention	***Description***
Notes	**Notes:** Notes are enclosed for emphasis
Bold	Bold type is used for emphasis, buttons, or tabs to select on windows, and names of dialog boxes.
TEXT IN SMALL CAPS	TEXT IN SMALL CAPS is used for keyboard keys that you press (for example, ALT+F4)
Text in italics	Italics are used to reference other documents, sections, figures, and tables.
`Special typestyle`	Special typestyle is used for filenames, directory names, commands, file listings, and anything that would appear in a Java program, such as methods, variables, and classes.
Bold italics typestyle	Bold italics is used for information the user needs to enter
{ }	Curly brackets are used for replaceable items (for example, replace {home directory} with its proper value such as C:\caadapter).

Table 1-1 Document text conventions

Chapter 2 caCORE Overview

This chapter provides an overview of the NCICB caCORE infrastructure.

Topics in this chapter include:

Architecture Overview

The NCI Center for Bioinformatics (NCICB) provides biomedical informatics support and integration capabilities to the cancer research community. NCICB has created a core infrastructure called Cancer Common Ontologic Representation Environment (caCORE), a data management framework designed for researchers who need to be able to navigate through a large number of data sources. By providing a common data management framework, caCORE helps streamline the informatics development throughout academic, government and private research labs and clinics. The components of caCORE support the semantic consistency, clarity, and comparability of biomedical research data and information. caCORE is open-source enterprise architecture for NCI-supported research information systems, built using formal techniques from the software engineering and computer science communities. The four characteristics of caCORE include:

- Model Driven Architecture (MDA)
- n-tier architecture with open Application Programming Interfaces (APIs)
- Use of controlled vocabularies, wherever possible
- Registered metadata

The use of MDA and *n*-tier architecture, both standard software engineering practices, allows for easy access to data, particularly by other applications. The use of controlled vocabularies and registered metadata, less common in conventional software practices, requires specialized tools, generally unavailable.

As a result, the NCICB (in cooperation with the NCI Office of Communications) has developed the Enterprise Vocabulary Services (EVS) system to supply controlled vocabularies, and the Cancer Data Standards Repository (caDSR) to provide a dynamic metadata registry. When all four development principles are addressed, the resulting system has several desirable properties. Systems with these properties are said to be "caCORE-like".

1. The *n*-tier architecture with its open APIs frees the end user (whether human or machine) from needing to understand the implementation details of the underlying data system to retrieve information.
2. The maintainer of the resource can move the data or change implementation details (Relational Database Management System, and so forth) without affecting the ability of remote systems to access the data.
3. Most importantly, the system is 'semantically interoperable'; that is, there exists runtime-retrievable information that can provide an explicit definition and complete data characteristics for each object and attribute that can be supplied by the data system.

Components of caCORE

The components that comprise caCORE are EVS, caDSR, caBIO, CSM, and CLM. Each is described briefly below.

Enterprise Vocabulary Services (EVS)

EVS provides controlled vocabulary resources that support the life sciences domain, implemented in a description logics framework. EVS vocabularies provide the semantic 'raw material' from which data elements, classes, and objects are constructed.

Cancer Data Standards Repository (caDSR)

The caDSR is a metadata registry, based upon the ISO/IEC 11179 standard, used to register the descriptive information needed to render cancer research data reusable and interoperable. The caBIO, EVS, and caDSR data classes are registered in the caDSR, as are the data elements on NCI-sponsored clinical trials case report forms.

Cancer Bioinformatics Infrastructure Objects (caBIO)

The caBIO model and architecture is the primary programmatic interface to caCORE. Each of the caBIO domain objects represents an entity found in biomedical research. Unified Modeling Language™ (UML) models of biomedical objects are implemented in Java as middleware connected to various cancer research databases to facilitate data integration and consistent representation. Examining the relationships between these objects can reveal biomedical knowledge that was previously buried in the various primary data sources.

Common Security Model (CSM)

CSM provides a flexible solution for application security and access control with three main functions:

- Authentication to validate and verify a user's credentials
- Authorization to grant or deny access to data, methods, and objects
- User Authorization Provisioning to allow an administrator to create and assign authorization roles and privileges.

Common Logging Module (CLM)

CLM provides a separate service under caCORE for Audit and Logging Capabilities. It also comes with a web based locator tool. It can be used by a client application directly, without the application using any other components like CSM.

In September of 2007, the NCI CBIIT Infrastructure and Product Management Team made the decision to separate the caCORE Components that had previously been bundled and released together. This decision was geared toward allowing each of the infrastructure product teams to be more responsive in addressing specific needs of the user community. With each component release there is a product specific Technical Guide. For more details on the other components, refer to the caCORE Overview page at http://ncicb.nci.nih.gov:80/NCICB/infrastructure/cacore_overview, which directs you to other product specific Technical Guides.

Chapter 3 caBIO Overview

This chapter provides an overview of the caBIO data and infrastructure.

Topics in this chapter include:

General Overview

The cancer Bioinformatics Infrastructure Objects (caBIO) model and architecture is a synthesis of software, vocabulary, and metadata models for cancer research. Each of the caBIO domain objects represents an entity found in biomedical research such as Gene, Chromosome, Nucleic Acid Sequence, SNP, Library, Clone, and Pathway. Examining the relationships between these objects can reveal biomedical knowledge that was previously buried in the various primary data sources. Given the dynamic nature of this information, the data in caBIO is updated on a monthly basis through a series of ETL (Extract, Transform, and Load) processes.

The caBIO data can be accessed via a variety of Application Programming Interfaces (APIs) as well as a web user interface. The system also provides a number of utilities to search for and manipulate data objects programmatically.

caBIO Data Overview

The caBIO system is primarily a repository of molecular biology data, and as such, the entities that concern the Central Dogma of Molecular Biology are the core of this model. Data for these classes come from a variety of sources. Gene, Chromosome, Nucleic Acid Sequence, and Taxon comes from the NCBI UniGene data resource both directly and via the Cancer Gene Anatomy Project (CGAP), an NCI activity that maintains information about expression of genes in tumors based on EST sequencing frequency. Cytogenetic location information, as well as EST and mRNA sequence annotation data come from the University of California Santa Cruz (UCSC) Genome Project. Protein data (including data about sequences) is retrieved from the Protein Information Resource (PIR) at Georgetown University.

The caBIO pathway data is primarily supplied from the BioCarta and its Proteomics Pathways Project via the NCI Cancer Molecular Analysis Project or CMAP. This pathway data is integrated into four primary classes: ‘Gene’, ‘Pathway’, ‘Taxon’, and ‘Histopathology’. The presence of the latter class is designed to allow a relationship between a pathway and a disease to be annotated. The Pathway object contains an attribute that contains an image of the pathway as a Scalar Vector Graphic (SVG).

Annotations from the most widely used human microarrays from Affymetrix (http://www.affymetrix.com), Illumina (http://www.illumina.com), and Aglient (http://www.agilent.com) are stored in caBIO, to allow the rich annotations provided by the manufacturers to be integrated with the other caBIO data sources and served through a caCORE interface. Data for these microarray annotations is distributed across many of the caBIO classes; the core classes are Microarray and ArrayReporter with subclasses SNPArrayReporter, ExpressionArrayReporter, and TranscriptArrayReporter.

Single Nucleotide Polymorphism data are primarily supplied by dbSNP and by Affymetrix (http://www.affymetrix.com). The core classes are SNP, PopulationFrequency, and GeneRelativeLocation.

The caBIO system provides information about libraries used as part of EST sequencing efforts. This set of classes (which includes Library, Tissue, and Protocol) are populated with data from the NCI CGAP initiative. This group of classes provides an instructive example about the necessity of grounding metadata in concept-based controlled terminology. The Protocol object here represents the method used to create a library from a tissue (NCI Thesaurus concept code C41111) rather than the concept of a clinical trial protocol (C25320).

Clinical trial data in the caBIO system are obtained from the Cancer Therapeutics Evaluation Program (CTEP) of the NCI. This part of the model was originally driven by use cases from the NCI CMAP project, and as such, the classes contain top level about ongoing clinical trials. In addition, the caBIO system provides some very basic information on the mechanism of action of drug compounds, variations found in disease and the genetic targets of drug compounds.

Architecture Overview

The caBIO system is an n-tier client-server architecture. The caBIO application server, located at NCI (http://cabioapi.nci.nih.gov/cabio40), provides a home page for web-based searching and data retrieval, as well as access to a wide variety of APIs for programmatic access.

The caBIO server interacts with a database backend on behalf of the user to serve requested objects to the client. Hibernate-based object-relational mapping (ORM) is used to populated caBIO domain objects. Only object-oriented APIs are exposed to clients so that the relational database backend is completely transparent to the user.

Many of the APIs rely on a query-by-example mechanism that takes a partially-populated object in, and returns other objects like it back to the user. This paradigm of interaction makes semantic interoperability much easier.

The following APIs are available for programmatic data access:

- Java API
- Web Services API
- REST API
- Grid Service

The system also provides several utilities for text searching and manipulation:

- Freestyle Lexical Mine (FreestyleLM) search
- XML Marshalling Utility for caBIO domain objects
- Pathway image manipulation utility

Chapter 4 caBIO Architecture

This chapter describes the architecture of the caBIO system. It includes information about the major components, such as database object-relational mappings (ORM), client-server communication, and how the system connects to non-ORM sources, such as FreestyleLM. It also describes the layout of the caBIO system packages. Topics in this chapter include:

caBIO System Architecture

The caBIO infrastructure exhibits an n-tiered architecture with client interfaces, server components, backend objects, data sources, and additional backend systems (Figure 4-1). This n-tiered system divides tasks or requests among different servers and data stores. This isolates the client from the details of where and how data is retrieved from different data stores. The system also performs common tasks such as logging and provides a level of security.

Clients (browsers, applications) receive information from backend objects. Java applications also communicate with backend objects via domain objects packaged within cabio40-beans.jar. Non-Java applications can communicate via SOAP (Simple Object Access Protocol). Back-end objects communicate directly with data sources, either relational databases (using Hibernate) or non-relational systems.

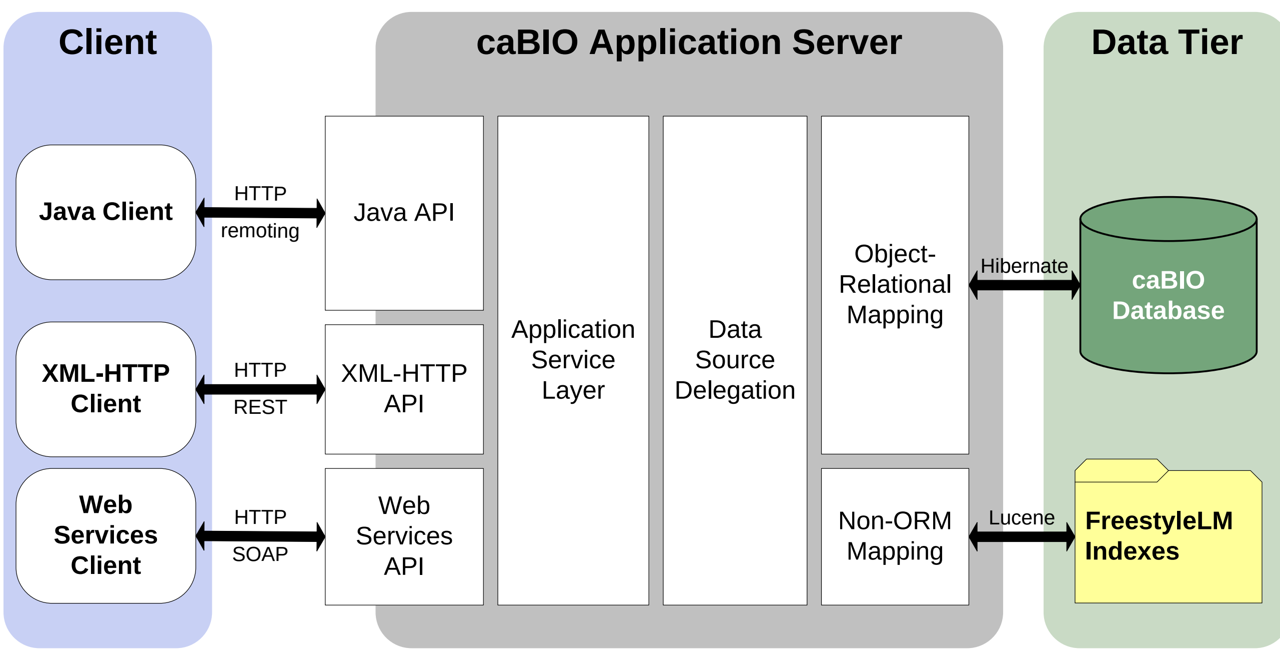


Figure 4-1 caBIO Architecture

Most of the caBIO infrastructure is written in the Java programming language and leverages reusable, third-party components.

The infrastructure is composed of the following layers.

The Application Service Layer

The Application Service layer consolidates incoming requests from various interfaces and translates them to native query requests that are then passed to the data layers. This layer is also responsible for handling client authentication and access control using the Java API. (This feature is currently disabled for the caBIO system running at NCICB; all interfaces provide full, anonymous read-only access to all data.)

The Data Source Delegation Layer

This layer is responsible for conveying each query that it receives to the respective data source that can perform the query. The presence of this layer enables multiple data sources to be exposed by a single running instance of a caBIO server.

Object-Relational Mapping (ORM)

The ORM mapping is implemented using Hibernate. Hibernate is a high performance object/relational persistence and query service for Java. Hibernate provides the ability to develop persistent classes following common object-oriented (OO) design methodologies such as association, inheritance, polymorphism, and composition.

The *Hibernate Query Language* (*HQL*), designed as a "minimal" object-oriented extension to SQL, provides a bridge between the object and relational databases. Hibernate allows for real world modeling of biological entities without creating complete SQL-based queries to represent them.

Non-ORM Layer

The caBIO database is a relational database. A Lucene-based document index is generated on the backend to facilitate the Lucene full-text search feature. The caBIO API implements the non-ORM based Lucene search as well as the ORM based Hibernate Lucene Search.

Client Technologies

Applications using the Java programming language can access caBIO directly through the domain objects provided by the cabio40-beans.jar (see Chapter 4 caBIO Architecture.) The network details of the communication to the caCORE servers are abstracted away from the developer. Hence, developers need not deal with issues such as network and database communication, but can instead concentrate on the biological problem domain.

The caBIO system also allows non-Java applications to use SOAP clients to interface with caBIO web services. SOAP is a lightweight XML-based protocol for the exchange of information in a decentralized, distributed environment. It consists of an envelope that describes the message and a framework for message transport. caBIO uses the open source Apache Axis package to provide SOAP-based web services to users. This allows other languages, such as Python or Perl to communicate with caBIO objects in a straightforward manner.

The caBIO architecture includes a presentation layer that uses a J2SE application server (such as Tomcat or JBoss). The JSPs (Java Server Pages) are web pages with Java embedded in the HTML to incorporate dynamic content in the page. caBIO also employs Java Servlets, which are server-side Java programs that web servers can run to generate content in response to client requests. All logic implemented by the presentation layer uses Java Beans, which are reusable software components that work with Java. All caBIO domain objects can be transformed into XML, the eXtensible Markup Language, as a universal format for structured data on the Web.

Communication between the client interfaces and the server components occurs over the Internet using the HTTP protocol. The server components are deployed in a web application container as a .war (Web archive) file that communicates with the back-end relational database management system that contains the actual data.

By using XSL/XSLT, the extensible stylesheet language for expressing stylesheets and XSL Transformations (XSLT), as a language for transforming XML documents, non-programmers can transform the information in the caBIO objects for use in documents or other systems.

Major caBIO Domain Packages

Table 4-1 shows the major caBIO domain packages from which you can access the Java interfaces and classes. All of the objects in the domain packages implement the java.io.Serializable interface. To view the JavaDocs page for each package, go to http://cabioapi.nci.nih.gov/cabio40/docs.

Package	***Description***
caBIO	Contains the domain-specific classes in the caBIO object package such as Gene, Chromosome, etc. For a list of the domain objects and their descriptions, see Chapter 6. `gov.nih.nci.cabio.domain`
Common	Contains a single class, DatabaseCrossReference, which is used by the caBIO domain package objects to provide links to related data hosted by other sources. Further details about this class are available in the next section and in Chapter 6. `gov.nih.nci.common.domain`
Provenance	Contains classes that provide provenance (source) metadata about the origin of data in caBIO objects. Further details about the classes in this package are available in Chapter 6. `gov.nih.nci.common.provenance.domain`

Table 4-1 caBIO packages and descriptions

The caBIO System Packages

Figure 4-2 displays a package for each application area in the caBIO API, described below.

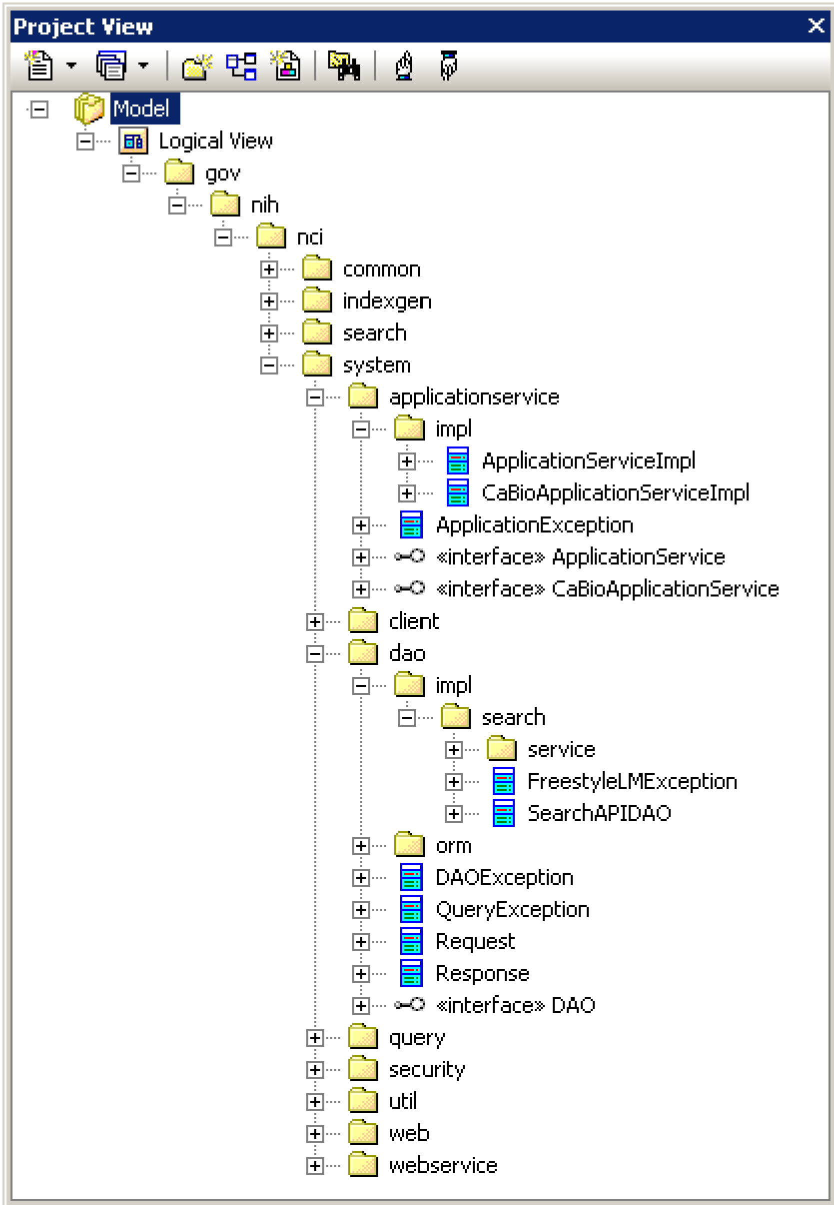


Figure 4-2 caBIO packages and descriptions

In addition to domain packages, the caBIO client API includes the following framework packages:

gov.nih.nci.common

The common package contains utility classes for manipulating caBIO domain objects.

gov.nih.nci.indexgen

The indexgen package contains the classes that generate the Lucene indexes for the caBIO database.

gov.nih.nci.search

The search package contains domain objects for the FreestyleLM search component and Grid Ids. Clients can instantiate these objects and pass them to appropriate API methods to search the FreestyleLM indexes or the Grid Ids.

gov.nih.nci.system

The system package has contains the client-side API, which communicates with the caBIO server, and helper classes for creating client-side proxies.

gov.nih.nci.system.client

The client package consists of classes that are used to generate dynamic proxies. The Application Service proxy classes intercept all the calls to the actual ApplicationService and takes action to facilitate the lazy-loading mechanism for the caBIO API.

gov.nih.nci.system.dao (Data Access)

The data access package is the layer at which the query is executed. This layer has implementation for querying ORM and non-ORM based data stores

gov.nih.nci.system.webservice

The Web service package contains the Web service wrapper class that uses Apache's Axis.

Chapter 5 Interacting with caBIO

This chapter describes the service interface layer provided by the caBIO architecture and gives examples of how to use each of the interface APIs.

Topics in this chapter include:

- caBIO Service Interface Paradigm on this page
- Java API on this page
- Web Services API on page 36
- XML-HTTP API on page 43
- FreestyleLM on page 47

caBIO Service Interface Paradigm

The caBIO API is built on top of the caCORE SDK. The caCORE architecture includes a service layer that provides a single, common access paradigm to clients using any of the provided interfaces. As an object-oriented middleware layer designed for flexible data access, caBIO relies heavily on strongly-typed objects and an object-in/object-out mechanism. The methodology used for obtaining data from the caBIO system is often referred to as query-by-example, meaning that the inputs to the query methods are themselves domain objects that provide the criteria for the returned data. The major benefit of this approach is that it allows for run-time semantic interoperability when used as part of the caCORE infrastructure, which provides shared vocabularies and a metadata registry.

The basic order of operations required to access and use the caBIO system is as follows:

1. Ensure that the client application has knowledge of the objects in the domain space.
2. Formulate the query criteria using the domain objects.
3. Establish a connection to the server.
4. Submit the query objects and specify the desired class of objects to be returned.
5. Use and manipulate the result set as desired.

There are four primary application programming interfaces (APIs) native to the caBIO system. Each of the available interfaces described below uses this same paradigm to provide access to the caBIO domain model, but with minor changes relating primarily to the syntax and structure of the clients. The following sections describe each API, identify installation and configuration requirements, and provide code examples.

Java API

The Java API provides direct access to domain objects and all service methods. Because caBIO is natively built in Java, this API provides the fullest set of features and capabilities.

Installation and Configuration

The caBIO Java API uses the following on the client machine (Table 5-1).

Software	*Version*	*Required?*
Java 2 Platform Standard Edition Software 5.0 Development Kit (JDK 5.0)	1.5.0 or higher	Yes
Apache Ant	1.6.5 or higher	Yes

Table 5-1 caBIO Java API Client software

Accessing the caBIO system also requires an Internet connection.

To use the Java API, download the client package provided on the NCICB web site (Figure 5-1).

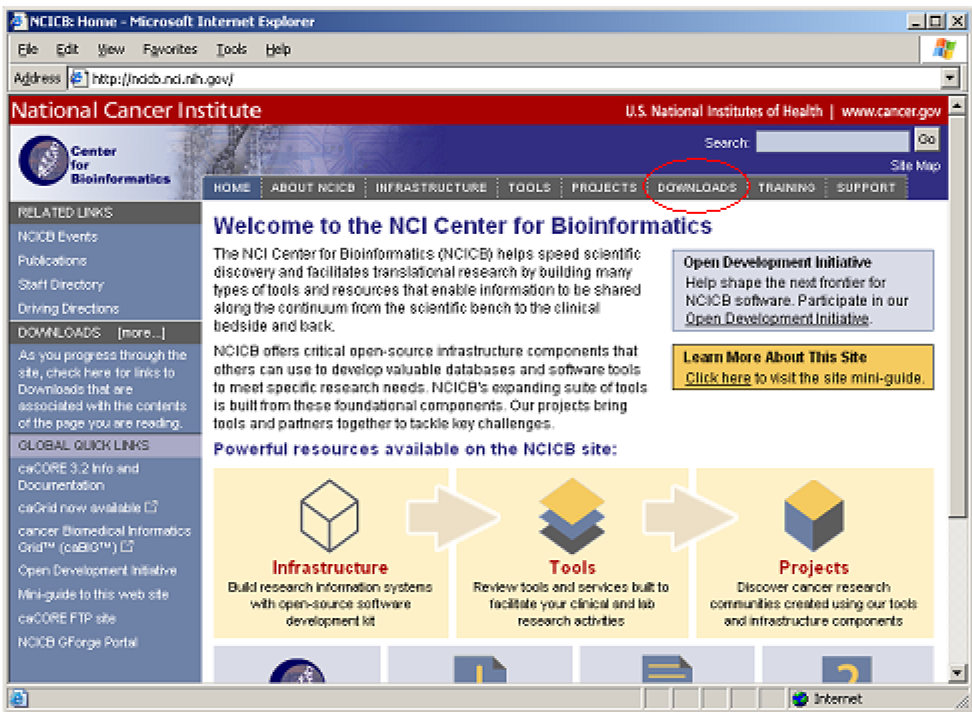


Figure 5-1 Downloads section on the NCICB website

1. Open your browser and go to http://ncicb.nci.nih.gov.
2. Click the Download link on the menu bar.
3. Scroll down to the section titled caBIO and click on the Download link.
4. In the provided form, enter your name, email address and institution name and click to Enter the Download Area.
5. Accept the license agreement.

6. On the caBIO downloads page, download the caBIO Zip file from the Primary Distribution section.
7. Extract the contents of the downloadable archive to a directory on your hard drive (for example, `c:\cabio` on Windows or `/usr/local/cabio` on Linux). The extracted directories and files include those in the following table (Table 5-2).

Directories and Files	Description	Component
build.xml	Ant build file	Build file
Src directory	Contains demo code	Sample code
TestClient.java	Java API client sample	SDK sample code
TestGetXMLClient.java	REST client sample	
TestXMLClient.java	XML marshalling sample	
TechGuideExamples.java	Java API examples from this guide	caBIO sample code
TestQueries.java	Java API query examples	
TestFreestyleLM.java	FreestyleLM search examples	
TestCQL.java	CQL API examples	
TestSVG.java	SVG pathway image examples	
TestXML.java	XML utility sample	
lib directory	contains required jar files	Java Libraries
cabio40-beans.jar	domain objects	caBIO API
cabio40-client-framework.jar	caBIO API extensions to the caCORE SDK Framework	
spring.jar	Spring framework	HTTP Remoting
hibernate*.jar	Hibernate	ORM layer
cglib-2.1.3.jar	CGLIB proxy library	
Log4j-1.2.13.jar	Log4j and Commons Logging	Logging Framework
commons-logging.jar		

Directories and Files	Description	Component
commons-discovery-0.2.jar		
castor-1.0.2.jar	Castor serializer/deserializer	XML conversion
xercesImpl.jar	Apache Xerces XML parser	
**.xsd*	XML schemas for objects	
handle.jar	Handle Service	Handle Service
handlerservice.jar	Custom handle code	
conf directory	Configuration files	
xml-mapping.xml	XML serialization mapping configuration files	XML mapping
unmarshaller-xml-mapping.xml		
application-config-client.xml	Spring configuration for client	Java API Spring configuration
Log4j.properties	Logging utilities configuration properties	Logging configuration

Table 5-2 Extracted directories and files in caBIO Java client package

All of the jar files provided in the lib directory of the caBIO client package in addition to the files in the conf directory are required to use the Java API. These should be included in the Java classpath when building applications. The `build.xml` file that is included demonstrates how to do this when using Ant for command-line builds. If you are using an integrated development environment (IDE) such as Eclipse, refer to the tool's documentation for information on how to set the classpath.

A Simple Example

To run the simple example program after installing the caBIO client, open a command prompt or terminal window from the directory where you extracted the downloaded archive and enter `ant rundemo`. This will compile and run the TechGuideExamples class; successfully running this example indicates that you have properly installed and configured the caCORE client. The following is a short segment of code from the TechGuideExamples class along with an explanation of its functioning.

```
ApplicationService appService = ApplicationServiceProvider.getApplicationService();
Gene gene = new Gene();
gene.setSymbol("brca*");  // searching for all genes whose symbols start with brca
try {
   List resultList = appService.search(Gene.class, gene);
   for (Iterator resultsIterator = resultList.iterator(); resultsIterator.hasNext();)
   {
      Gene returnedGene = (Gene) resultsIterator.next();
      System.out.println("Symbol: " + returnedGene.getSymbol() +
         "\tName " + returnedGene.getFullName()) +
         "\tTaxon:" + returnedGene.getTaxon().getScientificName();
      }
} catch (Exception e) {
     e.printStackTrace();
}
```

This code snippet creates an instance of a class that implements the ApplicationService interface. This interface defines the service methods used to access data objects. A criterion object is then created that defines the attribute values for which to search. The search method of the ApplicationService implementation is called with parameters that indicate the type of objects to return, `Gene.class`, and the criteria that returned objects must meet, defined by the gene object. The search method returns objects in a List collection, which is iterated through to print some basic information about the objects.

Although this is a simple example of the use of the Java API, a similar sequence can be followed with more complex criteria to perform sophisticated manipulation of the data provided by caBIO. Additional information and examples are provided in the sections that follow.

Service Methods

The methods that provide programmatic access to the caBIO server are located in the `gov.nih.nci.system.applicationservice` package. The `ApplicationServiceProvider` class uses the factory design pattern to return an implementation of the ApplicationService interface. The provider class determines whether there is a locally running instance of the caBIO system or whether it should use a remote instance. The returned ApplicationService implementation exposes the service methods that enable read/write operations on the domain objects.

Note: Although the infrastructure is capable of write operations, this functionality has been disabled for caBIO because it is primarily meant as a read-only data system.

The separation of the service methods from the domain classes is an important architectural decision that insulates the domain object space from the underlying service framework. As a result, new business methods can be added without needing to update any of the domain model or the associated metadata information from the object model. (This is critical for ensuring semantic interoperability over multiple iterations of architectural changes.)

Within the ApplicationService implementation, a variety of methods are provided allowing users to query data based on the specific needs and types of queries to be performed. In general, there are five types of searches:

- **Simple searches** are those that take one or more objects from the domain models as inputs and return a collection of objects from the data repositories that meet the criteria specified by the input objects.
- **Nested searches** also take domain objects as inputs but determine the type of objects in the result set by traversing a known path of associations from the domain model.
- **Detached criteria searches** use Hibernate detached criteria objects to provide a greater level of control over the results of a search (such as boolean operations, ranges of values, etc.)
- **HQL searches** provide the ability to use the Hibernate Query Language for the greatest flexibility in forming search criteria.
- **SDK Query Object criteria searches** were modeled similar to the Object representation of caBIG Query Language (CQL). The SDK Query Object criteria searches use syntax similar to the query by example (QBE) to specify the way results are to be retrieved. The system formulates the query based on the navigation path specified in the query search criteria. The query mechanism allows the user to search for the objects in platform independent query syntax.

Method Signature	`List search(`
	`Class targetClass,`
	`Object obj)`

Search Type	Simple (One criteria object)
Description	Returns a List collection containing objects of type targetClass that conform to the criteria defined by obj
Example	`search(Gene.class, gene)`

Method Signature	`List search(`
	`Class targetClass,`
	`List objList)`

Search Type	Simple (Criteria object collection)
Description	Returns a List collection containing objects of type targetClass that conform to the criteria defined by a collection of objects in objList. The returned objects must meet ANY criteria in objList (i.e. a logical OR is performed).
Example	`search(Gene.class, geneCollection)`

Method Signature	`List search(`
	`String path,`
	`Object obj)`

Search Type	Nested
Description	Returns a List collection containing objects conforming to the criteria defined by obj and whose resulting objects are of the type reached by traversing the node graph specified by path
Example	`search("gov.nih.nci.cabio.domain.Protein, gov.nih.nci.cabio.domain.Gene", nucleicAcidSequence)`

Method Signature	`List search(`
	`String path,`
	`List objList)`

Search Type	Nested
Description	Returns a List collection containing objects conforming to the criteria defined by the objects in objList and whose resulting objects are of the type reached by traversing the node graph specified by path
Example	`search("gov.nih.nci.cabio.domain.Protein, gov.nih.nci.cabio.domain.Gene", sequenceList)`

Method Signature	`List query(` `DetachedCriteria detachedCriteria)`
Search Type	Detached criteria
Description	Returns a List collection conforming to the criteria specified by detachedCriteria.

Example	`query(criteria)`

Method Signature	`List query(`
	`Object criteria,`
	`Integer firstRow,`
	`String targetClassName)`

Search Type	Detached criteria
Description	Identical to the previous query method, but allows for control over the size of the result set by specifying the row number of the first row.
Example	`query(criteria, 101, targetClassName)`

Method Signature	`List query(` `HQLCriteria hqlCriteria)`
Search Type	`Hibernate Query Language`
Description	Returns a List collection of objects that conform to the query in HQL syntax contained in hqlCriteria.
Example	`query(hqlCriteria)`

Method Signature	`List query(` `CQLQuery cqlQuery)`
Search Type	`Common Query Language`
Description	Returns a List collection of objects that conform to the query in SDK Query Object syntax contained in cqlQuery.
Example	`query(cqlQuery)`

In addition to the data access methods, the following helper methods are available via the ApplicationService class that provide flexibility in controlling queries and result sets.

Method Signature	`int getQueryRowCount(`
	`Object criteria,`
	`String targetClassName`

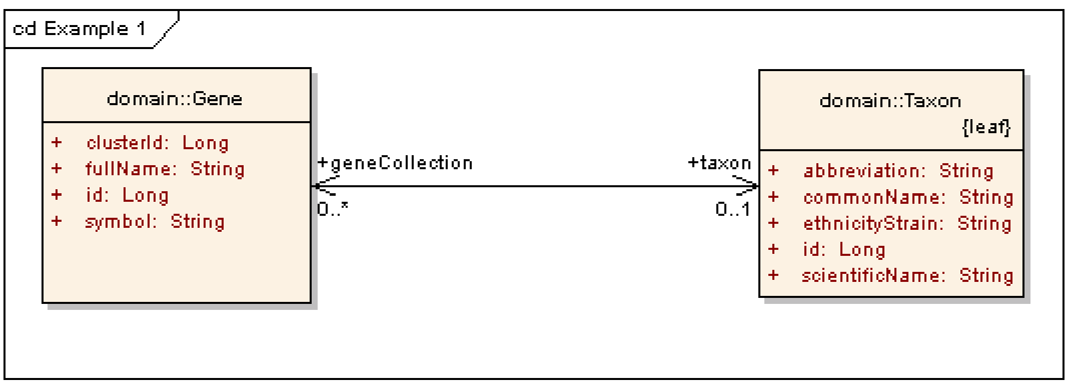
Description	Gets the number of objects in a given query. This is useful in determining the size of the data before querying for data
Example	`getQueryRowCount(criteria, "gov.nih.nci.cabio.domain.Gene")`

Examples of Use

This section includes a number of examples that demonstrate the use of the caBIO APIs. Included with each example is a brief description of the type of search being performed, a UML diagram depicting the domain objects used, and the example code accompanied by explanatory text.

Example One: Simple Search (Single Criteria Object)

In this example, a search is performed for all genes whose symbols start with 'brca'. The code iterates through the returned objects and prints out the symbol and name of each object along with the name of an associated object of type Taxon. The fetch of the associated Taxon object is done in the background and is completely transparent to the user.




```
ApplicationService appService = ApplicationServiceProvider.getApplicationService();
Gene gene = new Gene();
gene.setSymbol("brca*");  // searching for all genes whose symbols start with 'brca'
try
{
   List resultList = appService.search(Gene.class, gene);
   for (Iterator resultsIterator = resultList.iterator(); resultsIterator.hasNext();)
   {
      Gene returnedGene = (Gene) resultsIterator.next();
      System.out.println("Symbol: " + returnedGene.getSymbol() +
         "\tTaxon:" + returnedGene.getTaxon().getScientificName()
         "\tName " + returnedGene.getFullName());
   }
} catch (Exception e) {
     e.printStackTrace();
}
```

Lines	*Description*
1	Creates an instance of a class that implements the ApplicationService interface; this interface defines the service methods used to access data objects.
2-3	Creates a criterion object that defines the attribute values for which to query.
6	Calls the search method of the ApplicationService implementation and passes it the type of objects to return, Gene.class, and the criteria that returned objects must meet, defined by the gene object; the search method returns objects in a List collection.
9	Casts an object from the result List and creates a variable reference to it of type Gene.
10	Prints the symbol attribute.
11	Prints the fullName attribute.
12	Fetches an associated object of type Taxon and prints its scientificName attribute.

Example Two: Simple Search (Criteria Object Collection)

This example demonstrates a search with multiple criteria objects that are passed to the search() method. The result set will include all objects of the specified type that match ANY of the criteria objects. In this case, the search will return all objects of type Gene that are associated with Taxon objects whose abbreviation attribute is either "hs" or "ms". In biological terms, this search will return all human and mouse genes.

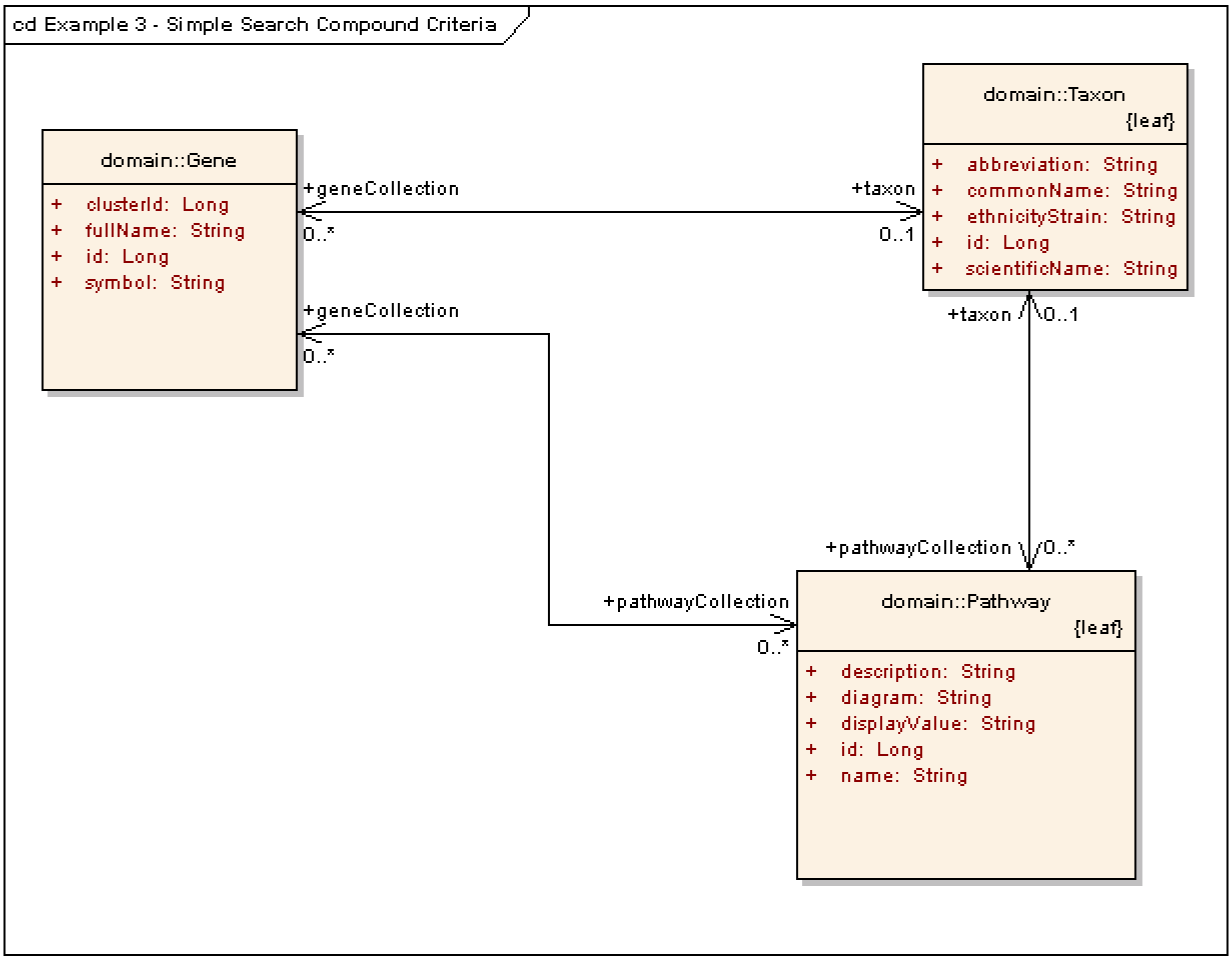
```
 1  Taxon taxon1 = new Taxon();
 2  taxon1.setAbbreviation("Hs");       // Homo sapiens
 3  Taxon taxon2 = new Taxon();
 4  taxon2.setAbbreviation("Mm");       // Mus musculus
 5  List<Taxon> taxonList = new ArrayList<Taxon>();
 6  taxonList.add(taxon1);
 7  taxonList.add(taxon2);
 8  try
 9  {
10     List resultList = appService.search(Gene.class, taxonList);
11     System.out.println("Number of results: "+resultList.size());
12  }
13  catch (Exception e) {
14     e.printStackTrace();
15  }
```

Lines	*Description*
1-4	Creates two Taxon objects describing the search criteria.
5-7	Creates a List collection containing the two Taxon objects.

10	Searches for all Gene objects where the associated Taxon object matches ANY of the objects found in the taxonList collection.

Example Three: Simple Search (Compound Criteria Object)

In this example, the object that is passed to the search() method contains criteria values that are found in associated objects and collections of objects. This query will return those objects that match all of the criteria in the compound object. Note the distinction between this type of search and the previous example in which a collection of objects is passed into the search method. In the last example, the results will match ANY of the criteria objects. In this example, however, where a single compound object is used, ALL criteria are matched. In biological terms, this search will return all pathways associated with the human Interleukin 5 gene.

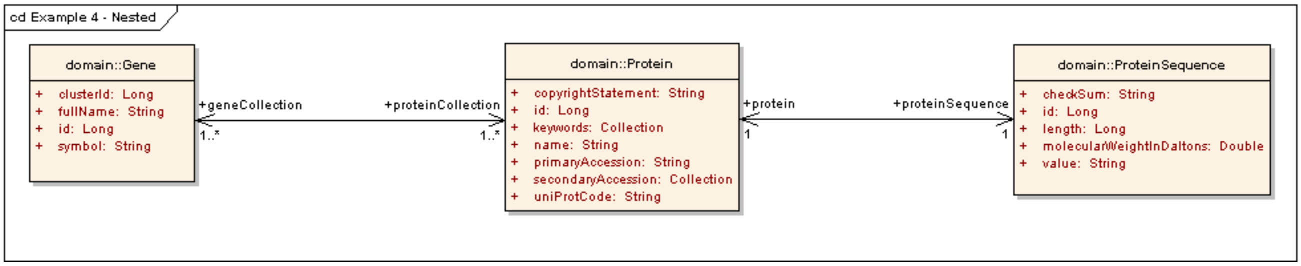


```
 1  Taxon taxon = new Taxon();
 2  taxon.setAbbreviation("hs");         // Homo sapiens
 3  Gene gene = new Gene();
 4  gene.setTaxon(taxon);
 5  gene.setSymbol("IL5");               // Interleukin 5
 6  List<Gene> geneList = new ArrayList<Gene>();
 7  geneList.add(gene);
 8  Pathway pathway = new Pathway();
 9  pathway.setGeneCollection(geneList);
10  try
11  {
12     List resultList = appService.search("gov.nih.nci.cabio.domain.Pathway", pathway);
13     for (Iterator resultsIterator = resultList.iterator(); resultsIterator.hasNext();)
14     {
15        Pathway returnedPathway = (Pathway)resultsIterator.next();
16        System.out.println("Name: "+returnedPathway.getName()
17            + "\tDisplayValue: " + returnedPathway.getDisplayValue());
18     }
19  } catch (Exception e) {
20     e.printStackTrace();
21  }
```

Lines	***Description***
1-5	Creates a Gene object and sets the symbol to "IL5" and the associated Taxon to an object whose abbreviation is set to "hs".
6-7	Because the Pathway and Gene classes are related by a many-to-many association, it is necessary to create a collection to contain the Gene object that will act as part of the compound criteria; multiple Gene objects could be added to this collection as needed.
8-9	Creates a Pathway object and sets the value of its geneCollection to the geneList object just created.
12	Searches for all Pathway objects whose geneCollection contains objects that match the set criteria (i.e. the symbol is "IL5" and the associated Taxon objects' abbreviations are set to "hs").

Example Four: Nested Search

A nested search is one where a traversal of more than one class-class association is required to obtain a set of result objects given the criteria object. This example demonstrates one such search in which the criteria object passed to the search method is of type *Gene*, and the desired objects are of type *ProteinSequence*. Because there is no direct association between these two classes, the path of the traversal is passed to the search method enabling the query to be performed.

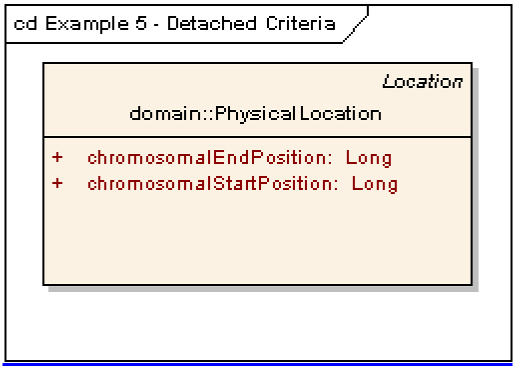



```
1   Gene gene = new Gene();
2   gene.setSymbol("TP53");              // Tumor protein p53 (Li-Fraumeni syndrome)
3   try
4   {
5      List resultList = appService.search(
6         "gov.nih.nci.cabio.domain.ProteinSequence,gov.nih.nci.cabio.domain.Protein",
7         gene);
8      for (Iterator resultsIterator = resultList.iterator(); resultsIterator.hasNext();)
9      {
10        ProteinSequence returnedProtSeq = (ProteinSequence)resultsIterator.next();
11        System.out.println("Id: " + returnedProtSeq.getId() +
12                           "Length: " + returnedProtSeq.getLength() );
13     }
14  } catch (Exception e) {
15     e.printStackTrace();
16  }
```

Lines	***Description***
1-2	Creates a Gene object and sets the symbol to "TP53".
6	Defines search path as traversing from the criteria object of type Gene through Protein to ProteinSequence; note that the first element in the path is the desired class of objects to be returned, and that subsequent elements traverse back to the criteria object.
7	Sets the criteria object to the previously-created Gene.

Example Five: Detached Criteria Search

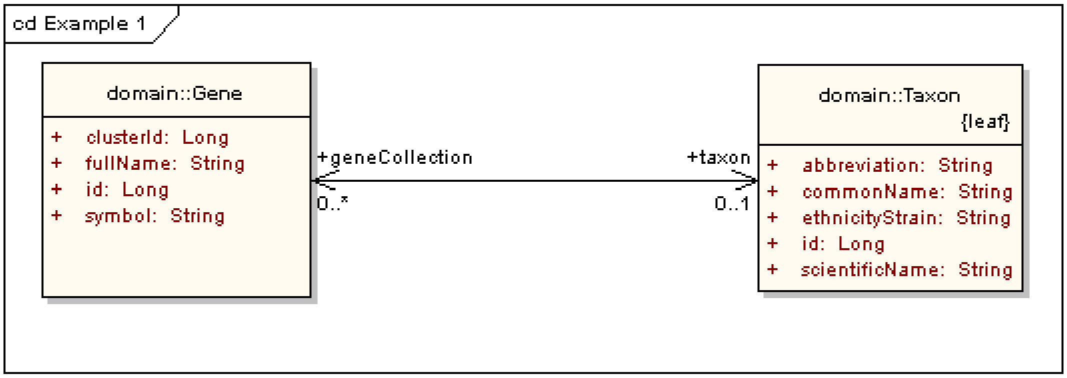
This example demonstrates the use of Hibernate detached criteria objects to formulate and perform more sophisticated searches. A detailed description of detached criteria is beyond the scope of this document; for more information, please consult the Hibernate documentation at http://www.hibernate.org/hib_docs/v3/api/org/hibernate/criterion/DetachedCriteria.html



Lines	Description
1	Creates a DetachedCriteria object and sets the class on which the criteria will operate to PhysicalLocation.
5	Sets a restriction on the objects that states that the attribute chromosomalStartPosition must be greater than ("gt") the value 86851632.
6	Sets a restriction on the objects that states that the attribute chromosomalEndPosition must be less than ("lt") the value 86861632.
7	Calls the query method of the ApplicationService implementation, passing the detached criteria object.

Example Six: HQL Search

In this example, a search is performed for all genes whose symbols start with 'brca'. This is identical to Example One but uses a Hibernate Query Language (HQL) search string to form the query. For more information on HQL syntax, consult the Hibernate documentation at http://www.hibernate.org/hib_docs/v3/reference/en/html/queryhql.html.




```
1 try {
2     String hqlString = "FROM Gene g WHERE g.symbol LIKE 'BRCA%'";
3     HQLCriteria hqlC = new HQLCriteria(hqlString);
4     List resultList = appService.query(hqlC);
5     System.out.println("\n Total # of records = " + resultList.size());
6 }
7 catch (Exception e) {
8     e.printStackTrace();
9 }
```

Lines	***Description***
2	Creates a string that contains the query in HQL syntax.
3	Instantiates an HQLCriteria object and sets the query string.
4	Calls the query method of the ApplicationService implementation and passes it the HQLCriteria object.

Example Seven: SDK Query Object Search

SDK Query Object Example One

```
1       ApplicationService appService =
                        ApplicationServiceProvider.getApplicationService();

2       CQLQuery cqlQuery = new CQLQuery();
3       CQLObject target = new CQLObject();  //Create Gene Object
4       target.setName("gov.nih.nci.cabio.domain.Gene");

5       CQLAttribute attribute = new CQLAttribute();
6       attribute.setName("symbol");
7       attribute.setValue("%il%");
8       attribute.setPredicate(CQLPredicate.LIKE);
9       target.setAttribute(attribute);

10      cqlQuery.setTarget(target);

11      try {
12              List resultList = appService.query(cqlQuery,
13                                              "gov.nih.nci.cabio.domain.Gene");
14              //Iterate through retrieved list of objects
15              for (Iterator resultsIterator = resultList.iterator();
16                                                      resultsIterator.hasNext();) {
17                      Gene returnedGene = (Gene) resultsIterator.next();
18                      System.out.println(
19                              "Symbol: " + returnedGene.getSymbol() + "\n" +
20                              "\tTaxon:" +
21                              returnedGene.getTaxon().getScientificName() +
22                              "\n" +"\tName " + returnedGene.getFullName() +
23                              "\n");
24              }
25      }
26      catch (Exception e) {
27              e.printStackTrace();
28      }
```

Lines	***Description***
2	Creates a new CQLQuery Object.
3-4	Creates a new search Target object (Gene).
5-8	Creates a new attribute for the search target object whose value should be compared with the value in the database for given predicate.
10	Sets the search target object in SDK Query Object Query.
12-13	Sets the search criteria object to the previously-created CQLQuery.

SDK Query Object Example Two

```
1       CQLQuery cqlQuery = new CQLQuery();
2       CQLObject target = new CQLObject();
3        target.setName("gov.nih.nci.cabio.domain.Gene");

4       CQLAttribute attribute = new CQLAttribute();
5       attribute.setName("symbol");
6       attribute.setValue("brca%");
7       attribute.setPredicate(CQLPredicate.LIKE);

8       //Create Taxon Object Query
9       CQLAssociation association1 = new CQLAssociation();
10      association1.setName("gov.nih.nci.cabio.domain.Taxon");
11      CQLAttribute attribute1 = new CQLAttribute();
12      attribute1.setName("abbreviation");
13      attribute1.setValue("%hs%");
14      attribute1.setPredicate(CQLPredicate.LIKE);
15      association1.setAttribute(attribute1);

16      //Create Taxon Object Query
17      CQLAssociation association2 = new CQLAssociation();
18      association2.setName("gov.nih.nci.cabio.domain.Taxon");
19      CQLAttribute attribute2 = new CQLAttribute();
20      attribute2.setName("abbreviation");
21      attribute2.setValue("%m%");
22      attribute2.setPredicate(CQLPredicate.LIKE);
23      association2.setAttribute(attribute2);

24      //Create Group(Collection) of Taxon Object Query
25      CQLGroup group = new CQLGroup();
26      group.addAssociation(association1);
27      group.addAssociation(association2);
28      group.setLogicOperator(CQLLogicalOperator.OR);
29      target.setAttribute(attribute);
30      target.setGroup(group);

31      cqlQuery.setTarget(target);

32      try {
33          List resultList = appService.query(cqlQuery);
34
35          //Iterate through retrieved list of objects
36          for (Iterator resultsIterator = resultList.iterator(); resultsIterator.hasNext();) {
37              Gene returnedGene = (Gene) resultsIterator.next();
38              System.out.println(
39                  "Symbol: " + returnedGene.getSymbol() + "\n" +
40                  "\tTaxon:" + returnedGene.getTaxon().getScientificName() +
41                  "\n" +"\tName " + returnedGene.getFullName() +"\n");
42          }
43      } catch (Exception e) {
44          e.printStackTrace();
45      }
```

Lines	***Description***
1	Creates a new CQLQuery Object.
2-3	Creates a new search Target object (Gene).
4-7	Sets attribute criteria for Target object
9-15, 17-23	Creates a new associated object (Taxon) whose attribute value should be compared with the value in the database for a given predicate.

Lines	**Description**
25-30	Creates a group (collection) of objects and associates it to the search target object.
31	Sets the search target object in SDK Query Object Query.
33	Search using the previously-created CQLQuery.

SDK Query Object Example Three

```
ApplicationService appService =
               ApplicationServiceProvider.getApplicationService();

CQLQuery cqlQuery = new CQLQuery();
CQLObject target = new CQLObject();
target.setName("gov.nih.nci.cabio.domain.Pathway");

//Create Gene Object Query
CQLAssociation association1 = new CQLAssociation();
association1.setName("gov.nih.nci.cabio.domain.Gene");
CQLAttribute attribute1 = new CQLAttribute();
attribute1.setName("symbol");
attribute1.setValue("IL5");
attribute1.setPredicate(CQLPredicate.EQUAL_TO);
association1.setAttribute(attribute1);

//Create Taxon Object Query
CQLAssociation association2 = new CQLAssociation();
association2.setName("gov.nih.nci.cabio.domain.Taxon");
CQLAttribute attribute2 = new CQLAttribute();
attribute2.setName("abbreviation");
attribute2.setValue("%hs%");
attribute2.setPredicate(CQLPredicate.LIKE);
association2.setAttribute(attribute2);

//Set Relatonship between Gene and Taxon
association1.setAssociation(association2);

//Set Relatonship between Pathway and Gene
//Role name is required as it can not be determined using
//reflection
association1.setTargetRoleName("geneCollection");
target.setAssociation(association1);

cqlQuery.setTarget(target);

try
{
       List resultList = appService.query(cqlQuery);
       //Iterate through retrieved list of objects
       for (Iterator resultsIterator = resultList.iterator();
                resultsIterator.hasNext();)
       {
               Pathway returnedPathway = (Pathway)resultsIterator.next();
               System.out.println(returnedPathway.getDisplayValue());
       }
} catch (Exception e) {
       e.printStackTrace();
}
```

Lines	***Description***
2	Creates a new CQLQuery Object.
3-4	Creates a new search Target object (Pathway).
5-12	Creates a new associated object (Gene) whose attribute value should be compared with the value in the database for a given predicate.
13-20	Creates a new associated object (Taxon) whose attribute value should be compared with the value in the database for a given predicate.
22	Creates a relationship between two objects (Gene and Taxon).
26-27	Creates a relationship between a target object (Pathway) and its associated object (Gene). The role name is the name of the association used to retrieve the associated object. It is required in case it can not be determined by reflection.
28	Sets the search target object in a SDK Query Object Query.
31	Sets the search criteria object to the previously-created CQLQuery.

SDK Query Object Example Four

```
1       ApplicationService appService =
                       ApplicationServiceProvider.getApplicationService();

2       CQLQuery cqlQuery = new CQLQuery();
3       CQLObject target = new CQLObject();
4       target.setName("gov.nih.nci.cabio.domain.ProteinSequence");

5       //Create Gene Object Query
6       CQLAssociation association1 = new CQLAssociation();
7       association1.setName("gov.nih.nci.cabio.domain.Gene");
8       CQLAttribute attribute1 = new CQLAttribute();
9       attribute1.setName("symbol");
10      attribute1.setValue("TP53");
11      attribute1.setPredicate(CQLPredicate.EQUAL_TO);
12      association1.setAttribute(attribute1);

13      //Create Protein Object Query
14      CQLAssociation association2 = new CQLAssociation();
15      association2.setName("gov.nih.nci.cabio.domain.Protein");

16      //Set Relatonship between Gene and Protein
17      //Role name is required as it can not be determined using
18      //reflection
19      association1.setTargetRoleName("geneCollection");
20      association2.setAssociation(association1);

21      //Set Relatonship between Protein and ProteinSequence
22      //Example of using relationship from target to source
24      association1.setSourceRoleName("proteinSequence");
25      target.setAssociation(association2);

26      cqlQuery.setTarget(target);

27      try
28      {
29          List resultList = appService.query(cqlQuery);
30          //Iterate through retrieved list of objects
31          for (Iterator resultsIterator = resultList.iterator();
32                  resultsIterator.hasNext();)
33          {
34              ProteinSequence returnedProtSeq =
35                  (ProteinSequence)resultsIterator.next();
36              System.out.println("ID: " + returnedProtSeq.getId() +
37                  "Length: " + returnedProtSeq.getLength());
38          }
39      } catch (Exception e) {
40              e.printStackTrace();
41      }
```

Lines	***Description***
2	Creates a new CQLQuery Object.
3-4	Creates a new search Target object (ProteinSequence).
5-12	Creates a new associated object (Gene) whose attribute value should be compared with the value in the database for a given predicate.
14-15	Creates a new associated object (Protein).

Lines	Description
19-20	Creates a relationship between a source object (Protein) and its associated object (Gene). The role name is the name of the association used to retrieve the associated object. It is required in case it can not be determined by reflection.
24-25	Creates a relationship between a target object (ProteinSequence) and its associated object (Protein).
26	Sets the search target object in a SDK Query Object Query.
29	Sets the search criteria object to the previously-created CQLQuery.

Utility Methods

XML Utility

The caCORE SDL provides a utility (*XMLUtility* class) in the `gov.nih.nci.common.util` package that provides the capability of converting caCORE domain objects between native Java objects and XML serializations based on standard XML schemas. The XML schemas for all domain objects in caCORE, directly generated from the UML model, are included in the downloadable archive (in the lib directory). Currently, the XML generated using the *XMLUtility* class includes only the object attributes; associated objects are not included.

Properties used by the XML utility are included in two files. The first, `xml.properties`, defines some basic information needed by the class and contains a property defining the filename of the second. This second file, called `xml-mapping.xml` by default, defines the binding between class, attribute names and the corresponding XML element, and attribute names.

A default marshaller and unmarshaller are provided with the caCORE client; developers wishing to use their own should provide the fully-qualified name of the two classes in the `xml.properties` file.

In the following code, the XML utility is used to serialize an object and save it to a file stream. A new object is then instantiated from the file using the utility.

```
1  // Assume an object of type Gene called myGene
2  File myFile = new File("myGene.xml");
3  FileWriter myWriter = new FileWriter(myFile);
4  XMLUtility myUtil = new XMLUtility();
5  myUtil.toXML(myGene,myWriter);
6  Gene myNewGene = (Gene)myUtil.fromXML(myFile);
7  bool isSameGene = myNewGene.equals(myGene);      // true
```

Lines	Description
2-3	Creates a new file stream where the XML serialization will be saved.

Lines	Description
4	Creates a new XMLUtility object; in this case, the default marshaller, and unmarshaller will be used.
5	Serializes the myGene object to XML using the mapping file and writes the output to the file stream myGene.xml.
6	Creates a new object called myNewGene by invoking the fromXML() method of the XMLUtility class and casting it to the proper type.
7	The newly created Gene object is equivalent to the old one.

For additional details, consult the caCORE JavaDocs at http://ncicb.nci.nih.gov/NCICB/content/ncicblfs/caCORE3-2_JavaDocs.

SVG Manipulation Utility

The caBIO API includes a class (*SVGManipulator*) in the `gov.nih.nci.common.util` package that provides useful services to manipulate Scalable Vector Graphics (SVG) diagrams retrieved from the caBIO Pathway domain object. For more information on how to use this utility, see SVG Manipulation Utility on page 36.

Web Services API

The caBIO Web services API allows access to caBIO data from development environments where the Java API cannot be used, or where use of XML Web services is more desirable. This includes non-Java platforms and languages such as Perl, C/C++, .NET Framework (C#, VB.Net), Python, etc.

The Web services interface can be used in any language-specific application that provides a mechanism for consuming XML Web services based on the Simple Object Access Protocol (SOAP). In these environments, connecting to caBIO can be as simple as providing the endpoint URL. Some platforms and languages require additional client-side code to handle the implementation of the SOAP envelope and the resolution of SOAP types. A list of packages catering to different programming languages is available at http://www.w3.org/TR/SOAP/ and at http://www.soapware.org/.

To maximize standards-based interoperability, the caBIO Web service conforms to the Web Services Interoperability Organization (WS-I) Basic Profile. The WS-I Basic Profile provides a set of non-proprietary specifications and implementation guidelines enabling interoperability between diverse systems. More information about WS-I compliance is available at http://www.ws-i.org.

On the server side, Apache Axis is used to provide SOAP-based inter-application communication. Axis provides the appropriate serialization and deserialization methods for the Java beans to achieve an application-independent interface. For more information about Axis, visit http://ws.apache.org/axis/.

Configuration

The caBIO WSDL file is located at http://cabioapi.nci.nih.gov/cabio40/services/caBIOService?wsdl.

In addition to describing the protocols, ports and operations exposed by the caBIO Web service, this file can be used by a number of IDEs and tools to generate stubs for caBIO objects. This allows code on different platforms to instantiate objects native to each for use as parameters and return values for the Web service methods. Consult the specific documentation for your platform for more information on how to use the WSDL file to generate class stubs.

The caBIO Web services interface has a single endpoint called caBIOService, which is located at http://cabioapi.nci.nih.gov/cabio40/services/caBIOService. Client applications should use this URL to invoke Web service methods.

Java WS Client

The Java WS client is a separate download from the Java API described earlier. For installation instructions, see the Java API on page 15. This distribution bundles the necessary Java libraries and example code that shows how to use the caBIO Web Service API.

Directories and Files	***Description***	***Component***
build.xml	Ant build file	Build
build.properties	Web Service configuration	Build
Src directory	Contains demo code	Sample code
TestClient.java	SDK WS client sample	
TestWS.java	Web services Java sample	
lib directory	contains required jar files	Java Libraries
cabio40-beans.jar	domain objects	caBIO API
hibernate-search.jar	Hibernate Search Annotations	ORM layer
axis.jar	Apache Axis	Web Services
saaj.jar	SOAP API for Java	
jaxrpc.jar	Java API for XML-based RPC	
wsdl4j-1.5.1.jar	WSDL for Java	
commons-logging.jar	Log4j and Commons Logging	Logging Framework

Directories and Files	***Description***	***Component***
commons-discovery-0.2.jar		
conf directory	Configuration files	
undeploy.wsdd	XML serialization mapping configuration files	XML mapping
application-config-client.xml	Spring configuration for client	Java API Spring configuration
Log4j.properties	Logging utilities configuration properties	Logging configuration

Table 5-3 Extracted directories and files in caBIO WS client package

All of the jar files provided in the lib directory of the caBIO client package in addition to the files in the conf directory are required to use the Java API. These should be included in the Java classpath when building applications. The `build.xml` file that is included demonstrates how to do this when using Ant for command-line builds. If you are using an integrated development environment (IDE) such as Eclipse, refer to the tool's documentation for information on how to set the classpath.

Operations

Through the caBIOService endpoint, developers have access to four operations:

Operation	queryObject
Input Schema	<complexType> <sequence> <element name="in0" type="xsd:string"/> <element name="in1" type="xsd:anyType"/> </sequence> </complexType>
Output Schema	<sequence> <element name="queryReturn"type= "ArrayOf_xsd_anyType"/> </sequence>
Description	Performs a search for objects conforming to the criteria defined by input parameter in1 and whose resulting objects are of the type reached by traversing the node graph specified by parameter `in0`; the result is a set of serialized objects (the type `ArrayOf_xsd_anyType` resolves to a sequence of `xsd:anyType` elements)

Operation	Query
Input Schema	<complexType> <sequence> <element name="in0" type="xsd:string"/> <element name="in1" type="xsd:anyType"/> <element name="in2" type="xsd:int"/> <element name="in3" type="xsd:int"/> </sequence> </complexType>

Output Schema	<sequence> <element name="queryReturn" type="ArrayOf_xsd_anyType"/> </sequence>
Description	Identical to the previous queryObject method, but allows for control over the result set by specifying the row number of the first row (in2) and the maximum number of objects to return (in3)

Developers should be aware of a significant behavioral decision that has been made regarding the Web services interface. When a query is performed with this interface, returned objects do not contain or refer to their associated objects. This means that a separate query invocation must be performed for each set of associated objects that need to be retrieved. One of the examples below demonstrates this functionality.

Examples of Use

Example One: Simple Search

The following code demonstrates a simple query written in the Java language that uses the Web services API. This example uses Apache Axis on the client side to handle the type mapping, SOAP encoding, and operation invocation.

```
1  Service  service = new Service();
2  Call call = (Call) service.createCall();
3
4  QName qnGene = new QName("urn:ws.domain.cabio.nci.nih.gov", "Gene");
5  call.registerTypeMapping(
6     Gene.class,
7     qnGene,
8     new org.apache.axis.encoding.ser.BeanSerializerFactory(Gene.class, qnGene),
9     new org.apache.axis.encoding.ser.BeanDeserializerFactory(Gene.class, qnGene)
10    );
11
12 String url = "http://cabioapi.nci.nih.gov/cabio40/services/caBIOService";
13
14 call.setTargetEndpointAddress(new java.net.URL(url));
15 call.setOperationName(new QName("caCOREService", "queryObject"));
16 call.addParameter("arg1", org.apache.axis.encoding.XMLType.XSD_STRING, ParameterMode.IN);
17 call.addParameter("arg2", org.apache.axis.encoding.XMLType.XSD_ANYTYPE, ParameterMode.IN);
18 call.setReturnType(org.apache.axis.encoding.XMLType.SOAP_ARRAY);
19
20 gov.nih.nci.cabio.domain.ws.Gene gene = new gov.nih.nci.cabio.domain.ws.Gene();
21 gene.setSymbol("IL*");
22
23 try
24 {
25    Object[] resultList = (Object[])call.invoke(
26       new Object[]{"gov.nih.nci.cabio.domain.ws.Gene", gene });
27    System.out.println("Total objects found: " + resultList.length);
28    if (resultList.length > 0)
29    {
30       for(int resultIndex = 0; resultIndex < resultList.length; resultIndex++)
31       {
32          Gene returnedGene = (Gene)resultList[resultIndex];
33          System.out.println(
34             "Symbol: " + returnedGene.getSymbol() + "\n" +
35             "\tName " + returnedGene.getFullName() + "\n" +
36             "");
37       }
38    }
39 } catch (Exception e) {
40    e.printStackTrace();
41 }
```

Lines	***Description***
1-2	Defines a new Web service call.
4-10	Maps a serialized object to its Java equivalent using the qualified name of the object from the WSDL file; in this case, the XML element Gene in the `urn:ws.domain.cabio.nci.nih.gov` namespace is mapped to the Java *Gene* class.
12	Defines the service endpoint.
14-18	Sets the call properties including the name of the operation to invoke, the input parameters that will be sent and the return type to expect.
20-21	Creates a Gene criteria object and sets its symbol attribute to `"IL*"`; note that the `*.ws.*` package is used.

Lines	Description
25-26	Invokes the Web service operation using an array of two objects (target class name and criteria object) as input parameters and expecting an object array as its result.
32	Casts each object in the result array to type Gene.

Example Two: Searching Associations

This example is similar to the previous one but demonstrates how to search for associated elements by performing additional invocations of the query or queryObject operation.

```
 1  try
 2  {
 3     Object[] resultList = (Object[])call.invoke(
 4        new Object[]{"gov.nih.nci.cabio.domain.Gene", gene });
 5     System.out.println("Total objects found: " + resultList.length);
 6     if (resultList.length > 0)
 7     {
 8        for(int resultIndex = 0; resultIndex < resultList.length; resultIndex++)
 9        {
10           Gene returnedGene = (Gene)resultList[resultIndex];
11           System.out.println(
12              "Symbol: " + returnedGene.getSymbol() + "\n" +
13              "\tName: " + returnedGene.getFullName() +
14              "");
15           Object[] nestedResultList = (Object[])call.invoke(
16              new Object[]{“gov.nih.nci.cabio.domain.ws.Taxon”, gene });
17           if (nestedResultList.length > 0)
18           {
19              Taxon returnedTaxon = (Taxon)nestedResultList[0];
20              System.out.println(“\tTaxon: “ + returnedTaxon.getScientificName());
21           }
22        }
23     }
24  } catch (Exception e) {
25     e.printStackTrace();
26  }
```

Lines	Description
15-16	A second operation invocation requests objects of type Taxon based on the same gene criteria used for the original query.
19	Casts the objects resulting from the nested query as Taxon objects.

Limitations

- By default, the queryObject operation limits the result set to 1000 objects, even if the size of the result set is larger. To retrieve the objects past the 1000th record, you must use the query operation and specify the first object index (parameter in2) to be greater than 1000.
- Result sets serialized and returned by the Web services interface do not currently include associations to related objects. A consequence of this behavior is that nested criteria objects with one-to-many associations that are passed to the query or

queryObject operations will result in an exception being thrown.

The following code demonstrates a Web services invocation that would fail:

```
Gene gene = new Gene();
gene.setSymbol("IL*");
Pathway pathway = new Pathway();
pathway.setId(new Long(120));
List pathwayList = new ArrayList()
pathwayList.add(pathway);
gene.setPathwaycollection(pathwayList);
try
{
   Object[] resultList = (Object[])call.invoke(
      new Object[]{"gov.nih.nci.cabio.domain.Gene", gene });
} catch (Exception e) {
   // Web Services Exception will be caught
}
```

Because the Web services invocation has an inherent timeout behavior, queries that take a long time to execute may not complete. If this is the case, use the query method to specify a smaller result count.

XML-HTTP API

The caCORE XML-HTTP API, based on the REST (Representational State Transfer) architectural style, provides a simple interface using the HTTP protocol. In addition to its ability to be invoked from most internet browsers, developers can use this interface to build applications that do not require any programming overhead other than an HTTP client. This is particularly useful for developing web applications using AJAX (asynchronous JavaScript and XML).

Service Location and Syntax

The caBIO XML-HTTP interface uses the following URL syntax as described in Table 5-4.

```
http://{server}/{servlet}?query={returnClass}&{criteria}&
resultCounter={counter}&startIndex={index}&
pageSize={pageSize}&pageNumber={pageNumber}
```

Element	***Meaning***	***Required***	***Example***
server	Name of the web server on which the caBIO web application is deployed.	Yes	cabio.nci.nih.gov
servlet	URI and the name of the servlet that will accept the HTTP GET requests	Yes	cabio40/server/GetXML cabio40/server/GetHTML

Element	Meaning	Required	Example
returnClass	Class name indicating the type of objects that this query should return	Yes	query=Gene
criteria	Search request criteria describing the requested objects	Yes	Gene[@id=2]
counter	Number of top level objects returned by the search	No	resultCounter=500
index	Start index of the result set	No	startIndex=25
pageSize	Number of records to display on each "page	No	pageSize=50
pageNumber	The number of the "page" for which to display results	No	pageNumber=3

Table 5-4 URL syntax used by the caBIO XML-HTTP interface

The caBIO architecture currently provides two servlets that accept incoming requests:

- **GetXML** returns results in an XML format that can be parsed and consumed by most programming languages and many document authoring and management tools
- **GetHTML** presents result using a simple HTML interface that can be viewed by most modern Internet browsers

Within the request string of the URL, the criteria element specifies the search criteria using XQuery-like syntax (Table 5-5). Within this syntax, square brackets ("[" and "]") represent attributes and associated roles of a class, the "at" symbol ("@") signals an attribute name/value pair, and a forward slash character ("/") specifies nested criteria. Criteria statements within XML-HTTP queries are generally of the following forms (although more complex statements can also be formed):

```
{ClassName}[@{attributeName}={value}] [@{attributeName}={value}]…
{ClassName}[@{attributeName}={value}]/{ClassName}[@{attributeName}={
value}]/…
```

Parameter	Meaning	Example
ClassName	The name of a class	Gene
attributeName	The name of an attribute of the return class or an associated class	symbol
Value	The value of an attribute	brca*

Table 5-5 Criteria statements within XML-HTTP queries

Examples of Use

The following examples demonstrate use of the XML-HTTP interface. In actual use, the queries shown here would either be submitted by a block of code or entered in the address bar of an Internet browser. Also note that the servlet name *GetXML* in each of the examples can be replaced with *GetHTML* to view with layout and markup in a browser.

Query	http://server/servlet/GetXML?query=Gene&Gene[@symbol=brca*]
Syntactic Meaning	Find all objects of type Gene whose symbol starts with 'brca'.
Biological Meaning	Find all BRCA genes.

Query	http://server/servlet/GetXML?query=Gene&Gene[@symbol=brca*]/Taxon[@scientificName=homo sapiens]
Syntactic Meaning	Find all objects of type Gene whose symbol starts with 'brca' and which have an associated Taxon object whose scientificName is equal to 'homo sapiens'.
Biological Meaning	Find all human BRCA genes.

Query	http://server/servlet/GetXML?query=Tissue&Tissue[@organ=eye][@histology=neoplasia]
Syntactic Meaning	Find all objects of type Tissue associated with attribute organ equal to 'eye' and histology equal to 'neoplasia'.
Biological Meaning	Find all tissues representing neoplasms of the eye.

Query	http://server/servlet/GetXML?query=Gene&Chromosome[@number=2]/Taxon[@scientificName=homo sapiens]
Syntactic Meaning	Find all objects of type Gene associated with Chromosome objects with number equal to 2 which themselves are related to Taxon objects with scientificName equal to 'homo sapiens'.

Biological Meaning	Find all human genes on chromosome number 2

Query	http://server/servlet/GetXML?query=Gene&Chromosome[@number=2]/Taxon[@scientificName=homo sapiens]
Syntactic Meaning	Find all objects of type Gene associated with Chromosome objects with number equal to 2 which themselves are related to Taxon objects with scientificName equal to 'homo sapiens' in biological terms
Biological Meaning	Find all human genes on chromosome number 2

Working With Result Sets

Because HTTP is a stateless protocol, the caBIO server has no knowledge of the context of any incoming request. Consequently, each invocation of GetXML or GetHTML must contain all of the information necessary to retrieve the request, regardless of previous requests. Developers should consider this when working with the XML-HTTP interface.

Retrieving Related Results using XLinks

When using the GetXML servlet to retrieve results as XML, associations between objects are converted to XLinks within the XML. The link notation, shown below, allows the client to make a subsequent request to retrieve the associated objects.

```
<class name="gov.nih.nci.cabio.domain.Gene" recordNumber="1">
   …
   <field name="taxon"
      xlink:type="simple"
      xlink:href=
"http://cabioapi.nci.nih.gov/cabio40/GetXML?query=Taxon&Gene[@id=5]"
>
         getTaxon
      </field>
   …
</class>
```

Controlling the Number of Items Returned

The GetXML servlet provides a throttling mechanism to allow developers to define the number of results returned on any single request and where in the result set to start. For example, if a search request yields 500 results, specifying resultCounter=450 will return only the last 50 records. Similarly, specifying startIndex=50 will return only the first 50 records.

Paging Results

In addition to controlling the number of results to display, the GetXML servlet also provides a mechanism to support "paging". This concept, common to many web sites, allows results to be displayed over a number of pages, so that, for example, a request that yields 500 objects could be displayed over 10 pages of 50 objects each. When the paging feature is used, the GetXML servlet will include XLinks to each of the result pages in an XML <page/> element. The element data of the <page/> element is the number of the page, suitable for output as text or HTML when using an XSL stylesheet:

```
<page number="1"
   xlink:type="simple"
   xlink:href="http://cabioapi.nci.nih.gov/cabio40/GetXML?query=
{query}&pageNumber=4&resultCounter=1000&startIndex=0"> 4 </page>
```

Limitations

When specifying attribute values in the query string, use of the following characters generates an error: [ ] / \ # & %.

FreestyleLM

All of the search methods discussed so far in this chapter allow for *vertical* searching in one caBIO domain object. In other words, one domain object must be selected to be searched. The FreestyleLM (Freestyle Lexical Mine) search component, new in 4.0, provides and interfaces and APIs for conducting *horizontal* searches across caBIO domain objects.

FreestyleLM is build upon Hibernate and Apache Lucene full text search engines, providing full text search (Google™-like search) capabilities to the caBIO API. The classes and fields that are indexed are annotated in the caBIO domain model. Usually, these attributes are non-numeric fields with descriptions or identifiers. Thus, searching for “brca1” will return any object that makes mention of the “brca1” gene in its attributes, such as Genes, Proteins, Pathways, and so on.

The IndexGenerator utility within the caBIO API generates index files for these annotations. FreestyleLM provides the ability to query these indexes. The FreestyleLM search is facilitated by the classes listed below:

gov.nih.nci.search.SearchQuery
gov.nih.nci.search.SearchResults
gov.nih.nci.search.Sort
gov.nih.nci.search.RangeFilter

The search term is set as a keyword within the SearchQuery object. One or more words can be set as a keyword. Refer to the Lucene documentation for more information on the search syntax (http://lucene.apache.org/java/docs/queryparsersyntax.html).

Freestyle LM Java Client

The CaBioApplicationService provides a special data access method to perform the text based search.

Method Signature	`List search(SearchQuery searchQuery)`
Search Type	FreestyleLM Search
Description	Returns a List collection containing objects conforming to the criteria defined by the searchQuery.
Example	`Search(searchQuery);`

Example One: Full Text Search (Default)

In this example, a search is performed on the term ‘brca*’. The caBIO FreestyleLM returns a set of SearchResults objects. The result set will include reference to objects of different types that match the search term. In this case, the search results consist of information regarding Gene, Protein, Pathway, Tissue, GeneOntology and Library objects.

```
1 CaBioApplicationService appService =
      (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();
2 try {
3     SearchQuery query = new SearchQuery();
4     query.setKeyword("brca*");
5     List results = appService.search(query);
6     for(int i=0; i<results.size(); i++) {
7         SearchResult result = (SearchResult)results.get(i);
8         System.out.println("Class: "+ result.getClassName() +"\t"+ result.getId());
9     }
10    System.out.println("Results: "+ results.size());
11 } catch(Exception e) {
12      e.printStackTrace();
13 }
```

Lines	***Description***
1	Create a CaBioApplicationService instance.
3-4	Instantiate a SearchQuery object and set the keyword to ‘brca*’
5	Call the CaBioApplicationService search method
6-9	Iterate through the results

Example Two: Hibernate Search

A search is performed on the term ‘brca*’ and the query type is specified as ‘HIBERNATE_SEARCH’. (The default query type is FULL_TEXT_SEARCH). The caBIO FreestyleLM result set will include objects of different types that match the search term. In this case, the search results consist of Gene, Protein, Pathway, Tissue, GeneOntology and Library objects.

```
CaBioApplicationService appService =
      (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();
try {
    SearchQuery query = new SearchQuery();
    query.setKeyword("brca*");
    query.setQueryType("HIBERNATE_SEARCH");
    List results = appService.search(query);
    for(int i=0; i<results.size(); i++){
         Object result = (Object)results.get(i);
         System.out.println("Class: "+ result.getClass().getName());
    }
    System.out.println("Results: "+ results.size());
} catch(Exception e){
    e.printStackTrace();
}
```

Example Three: Sort Results

The result set can be sorted by specifying a sort value in the SearchQuery.

```
CaBioApplicationService appService =
     (CaBioApplicationService)ApplicationServiceProvider.getApplicationService();
try {
    SearchQuery query = new SearchQuery();
    query.setKeyword("brca*");
    Sort sorter = new Sort();
    sorter.setSortByClassName(true);
    query.setSort(sorter);
    List results = appService.search(query);
    for(int i=0; i<results.size(); i++){
        SearchResult result = (SearchResult)results.get(i);
        System.out.println("Class: "+ result.getClassName() +"\t"+ result.getId());
    }
    System.out.println("Results: "+ results.size());
} catch(Exception e) {
    e.printStackTrace();
}
```

Freestyle LM Web Interface

The caBIO API provides a simple web interface to perform text based queries. This user interface is built upon the FreestyleLM Search API (Figure 5-2). The following URL can be used to access the servlet:

http://cabioapi.nci.nih.gov/cabio40/search

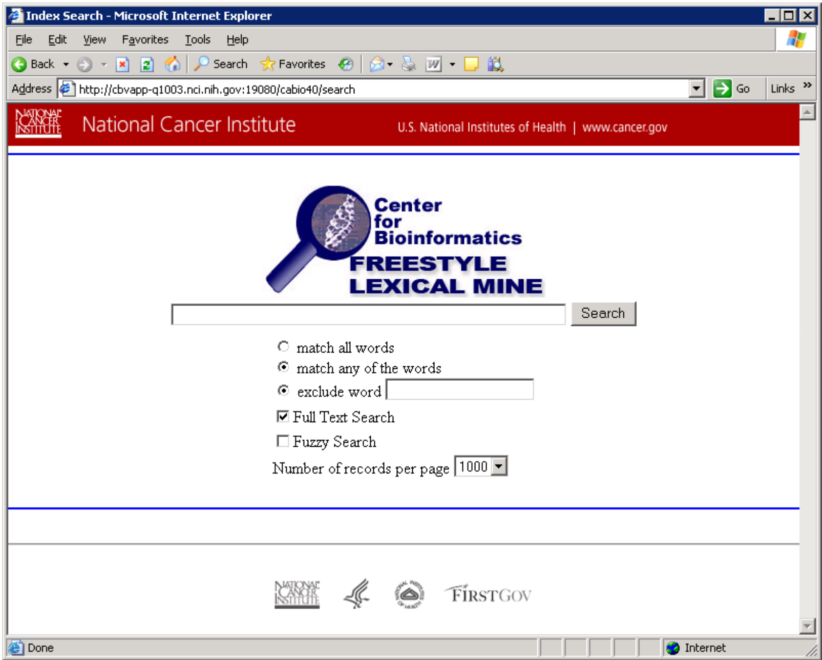


Figure 5-2 Web interface for performing text based queries

Chapter 6 Cancer Bioinformatics Infrastructure Objects

This chapter describes the Cancer Bioinformatics Infrastructure Objects (caBIO) model and its application programming interfaces.

Topics in this chapter include:

Introduction

The Cancer Bioinformatics Infrastructure Objects (caBIO) model and architecture was the first of several model-driven information systems that make up caCORE and continues to be an on-going effort to model the genomic domain. The caBIO objects simulate the behavior of actual genomic components such as genes, chromosomes, sequences, libraries, clones, ontologies, etc. They provide access to a variety of genomic data sources including GenBank, Unigene, LocusLink, Homologene, Ensemble, UCSC Genome Sequencing Center's Annotations, NCI's CTEP (Cancer Therapy Evaluation Program) and NCICB's CGAP (Cancer Genome Anatomy Project) data repositories. The full list of data sources is listed starting on page 58.

caBIO API

Most of the domain objects defined in the caBIO API are objects that specialize in bioinformatics applications. The caBIO domain objects are implemented as Java beans in the gov.nih.nci.cabio.domain, gov.nih.nci.common.domain and gov.nih.nci.common.provenance.domain packages and include those classes that correspond to biological entities and bioinformatic concepts.

The caBIO UML model is published as an EA (Enterprise Architect) diagram at http://ncicb.nci.nih.gov/NCICB/content/ncicblfs/EA/caCORE4-0Model/index.htm. Table 6.1 and 6.2 list the classes in these packages along with a description. Detailed descriptions about each class and its methods are present in the caCORE 4.0 JavaDocs.

caBIO Domain Object	***Description***
Agent	An object representing a therapeutic agent (drug, intervention therapy) used in a clinical trial protocol. It provides the NSC Number, name and source of the agent. An *Agent* provides access to the associated protocols used in clinical trials and associated genes targeted (*Targets*) by that agent in the clinical trials via the *getClinicalTrialProtocolCollection* and *getTargetCollection* functionalities.

caBIO Domain Object	Description
Anomaly	An irregularity in either the expression of a gene or its structure (i.e., a mutation). It provides access to the associated histopathological information (*Histology*) as well as the disease (*DiseaseOntology*) and organ ontology *(OrganOntology)*. A user can also retrieve the anomalies *(Anomaly)* associated with a given gene target using the *getAnomalyCollection* functionality.
ArrayReporter	A superclass for different kinds of array reporters such as *ExonArrayReporter*, *SNPArrayReporter*, *TranscriptArrayReporter* and *ExpressionArrrayReporter*. These objects are populated with annotations from Affymetrix, Illumina and Agilent Arrays such as the Human Mapping Arrays, the HG-U133 Plus2 Arrays, the Exon Arrays, HumanHap Arrays, etc.
Chromosome	An object representing a specific chromosome on a human or mouse; It provides access to all known genes on the chromosome, as well as to the *Location* of features such as SNP, ESTs, mRNAs, etc through the *getLocationCollection* functionality. The SNP, EST and MRNA Annotations are obtained from NCBI's dbSNP and UCSC's Genome Sequencing Center respectively.
ClinicalTrialProtocol	The protocols and administrative information about the trial such as lead organization, participants, current phase of the clinical trial, and associated NIH Administrative Id. A *ClinicalTrialProtocol* object provides the user access to the associated agents, histopathology and disease ontologies through the *getAgentCollection*, *getDiseaseOntologyCollection* and *getHistopathologyCollection* functionalities respectively. This data is obtained from NCI's CTEP Program.
Clone	An object used to hold information pertaining to I.M.A.G.E. clones associated with human and mouse genes from Unigene's repository. A user can access the associated sequences (*NucleicAcidSequence)* using the *getNucleicAcidSequenceCollection* functionality. A clone is associated with a clone-library further details of which are available through the associated *Library* Object. The orientation of the clone along with the associated sequence identifier is available through the *CloneRelativeLocation* object that provides further information regarding the clone insert read.
CloneRelativeLocation	Provides the orientation (5'/3') of a clone insert read, along with links to the associated sequence information (*NucleicAcidSequence*).

caBIO Domain Object	***Description***
Cytoband	The *Cyotband* Object provides information such as the name and location of cytobands in human and mouse available from UCSC's Genome Sequencing Center for every new build of the Human and Mouse Genome. The positions of cytogenetic bands within the human or mouse chromosomes are available through the associated *CytogeneticLocation* object. Additionally, it also provides an interface to the cytoband information available as part of Affymetrix Human Mapping Arrays, if any.
CytogeneticLocation	Encapsulates the location information for the *Cytoband*s with associated *SNP*s and *Gene*s.
DatabaseCrossReference	An object that cross-references entities such as *Gene*s, *SNP*s, etc. from different databanks. For example, it maps gene Ids from Unigene with those from Entrez-Gene/Locus-Link, OMIM, Ensembl and Enzyme Consortium databanks. It cross-references *SNP*s from NCBI's dbSNP with those from The SNP Consortium (TSC) and proteins from *Uniprot* with *RefSeq*.
DiseaseOntology	An object representing various oncological disease ontologies. It provides access to the clinical trials associated with these diseases (*ClinicalTrialProtocol*) and the associated histopathological (*Histopathology*).
DiseaseOntology Relationship	Provides a hierarchical relationship among various diseases and associated ontologies. Thus it allows a user to obtain *DiseaseOntology*s that share a parent/child relationship
Exon	An *Exon* represents an expressed part of a gene. In caBIO, an exon additionally constitutes part of a *Transcript* used as a probe in Affymetrix Exon Arrays. It provides access to associated *ExonArrayReporters* and to the *Transcripts* to which it is a part.
ExonArrayReporter	One of the two 'transcript-array' objects in caBIO, the other being *ExpressionArrayReporter*. An object representing Affymetrix Exon Arrays providing details such as the manufacturer's PSR Id, the probe count and the strand information for each probeset. A user can retrieve the associated *Exons*, *Transcript*s and *Genes* using the *getTranscriptCollection* and *getGeneCollection* methods respectively.
ExpressionArrayReporter	One of the two objects representing transcript-based arrays in caBIO. It provides information on probesets such as the type and source of the transcripts in arrays such as the Affymetrix Human

caBIO Domain Object	Description
	U133 Plus2 Arrays, and Agilent (Human-Genome 44k and a-CGH 244K) oligo Arrays. A user can retrieve the associated *Gene*s and protein domains (*ProteinDomain*) through the *getGeneCollection* and *getProteinDomainCollection* methods respectively. The sequence information for the gene can be obtained from *NucleicAcidSequence* using *getNucleicAcidSequenceCollection* method.
Gene	Gene objects are the effective portal to most of the genomic information provided by the caBIO data services; It provides access to the associated proteins, gene-targets, diseases, histopathologies, ontologies, pathways, aliases, clones and sequence data through their respective objects such as *Histopathology, DiseaseOntology, OrganOntology, Pathway, GeneAlias, Clone* and *NucleicAcidSequence* respectively. The location of the gene on the chromosome is available through the *GeneRelativeLocation and Location* objects. This object is populated with data pertaining to human and mouse from Unigene's repository. Corresponding Genes from NCBI's OMIM Databank, Ensembl, and Enzyme Commission are available through *DatabaseCrossReference*.
GeneAlias	An alternative name for a gene; provides descriptive information about the gene (as it is known by this alias), as well as access to the primary *Gene* it refers to.
GeneOntology	An object providing information on a *Gene*'s position in the Gene Ontology Consortium's controlled vocabularies. It provides access to *Gene*s corresponding to an ontological term and to the relevant parent/child ontological relationship, if any via the corresponding *GeneOntologyRelationship*.
GeneOntology Relationship	Specifies the parent/child relationship, if any, for a given *Gene Ontology* and allows the user to retrieve such associated *GeneOntology*s.
GeneRelativeLocation	An object providing relative location of the *Gene*s and *SNP*s, in microarray probesets from the Affymetrix Human Mapping and Illumina 550K genotyping arrays.
GenericArray	Represents the physical chip along with its features and the features' annotations. Currently represents only the Affymetrix arrays. Deprecated from this release as this, along with additional data is exposed through MicroArray object.
GenericReporter	A deprecated version superseded by ArrayReporter. It exposes only Affymetrix HG-U133 Plus2 arrays (SNP Arrays) and will no longer be available in the next release

caBIO Domain Object	***Description***
Histopathology	Represents anatomical changes in a diseased tissue sample associated with an expression experiment; captures the relationship between an organ and disease. A user can also retrieve any associated *Pathway*s.
Homologous Association	Represents degrees of homology (as a percentage) amongst various genes in human and mouse, along with access to further details about the *Genes* themselves. It is populated with human and mouse data from NCBI's HomoloGene Project.
Library	An object representing a CGAP library; provides access to information about: the tissue sample and its method of preparation, the library protocol that was used, the clones contained in the library, and the sequence information derived from the library.
Location	Super class for *PhysicalLocation* and *CytogenicLocation* objects.
Marker	An object representing the Microsatellite marker in Affymetrix HuMapping Arrays. It provides information such as the type of marker and the associated *SNP*s from The SNP Consortium. The SNP's relative location is available through *MarkerRelativeLocation*.
MarkerRelativeLocation	An object providing location information for microsatellite *Markers*. The associated probeset information is available through *SNPReporter*
MicroArray	An object representing various microrrays in caBIO such as the Affymetrix Human Mapping, HG-U133 Plus2 and Illumina HumanHap 550K 'SNP'/genotyping arrays, Affmetrix 'Exon' Arrays and 'oligo' arrays such as the Agilent Human Genome 44k and aCGH 244k arrays.
NucleicAcidSequence	An object representation of a gene sequence; provides access to the clones from which it was derived, the ASCII representation of the residues it contains, and the sequence ID and a link to the Location Objects that in turn provide further details of any SNP, Cytoband, EST or MRNAs associated with that sequence. It is populated with data pertaining to human and mouse from Unigene.
OrganOntology	A representation of an organ whose name occurs in a controlled vocabulary; provides access to any associated histopathological information, and anomalies. Because it is "ontolog-able", It provides access to any associated ontologies that it shares a parent/child relationship with, through the linked

caBIO Domain Object	Description
	OrganOntologyRelationship.
OrganOntologyRelationship	An object that describes parent/child relationships, if any, amongst *OrganOntology*s.
Pathway	An object representation of a molecular/cellular pathway in human and mouse compiled by BioCarta. It provides access to further details about the associated *Gene*s and any available histopathological information for genes involved in clinical trials through *getGeneCollection* and *getHistopathologyCollection* methods.
PhysicalLocation	Provides chromosomal start and stop positions for *SNP*s, ESTs, MRNAs and Cytobands. Further details for the SNP can be obtained from the *SNP* object. The sequence associated with the ESTs and MRNAs can be obtained from the linked *NucleicAcidSequence* object. The SNP Data is obtained from NCBI's dbSNP, while the latter are obtained from UCSC's Genome Sequencing Center for every Human and Mouse Genome build.
PopulationFrequency	Represents the major and minor alleles of a SNP and their respective frequencies in different populations.
Protein	An object representation of a protein; provides access to the encoding Gene via its GenBank ID, the Taxon in which this instance of the protein occurs, and references to homologous proteins in other species. It is populated with data from Uniprot-Swissprot.
ProteinAlias	An alternate name for a protein. This data is obtained from Uniprot-Swissprot.
ProteinDomain	An object representing protein domains from Interpro. This object is populated with data from Affymetrix HG-U133 Plus2 Array annotations.
ProteinSequence	The sequence of a protein from Swissprot.
Protocol	An object representation of the protocol used in assembling a clone library. It is populated with data from NCI's CTEP (Cancer Therapy Evaluation Program)
ProtocolAssociation	An association class relating *ClinicalTrialProtocols* to *Diseases*.
RelativeLocation	An abstract super class for *MarkerRelativeLocation* and *GeneRelativeLocation* that provides the relative locations of microsatellite markers, genes and SNPs associated with different MicroArray probesets in caBIO.

caBIO Domain Object	***Description***
SNP	An object representing a Single Nucleotide Polymorphism; It provides the two most common substitutions at that position and the offset of the SNP in the parent sequence. A reference to related SNPs from The SNP Consortium (TSC) is available through DatabaseCrossReference. It provides a link to the associated Population Frequencies. The location of a SNP is available through the *PhysicalLocation* and *Location* Objects as well.
SNPArrayReporter	An object representing probeset information for genotyping arrays such as Affymetrix Human Mapping Arrays and Illumina HumanHap 550k Arrays; It provides details on the corresponding SNP through the linked *SNP* object.
Target	A gene thought to be at the root of a disease etiology, and which is targeted for therapeutic intervention in a clinical trial.
Taxon	An object representing the various names (scientific, common, abbreviated, etc.) for human and mouse genomes. It can be used to retrieve the associated Genes, Chromosomes, Pathways if any, Proteins, or Tissues.
Tissue	A group of similar cells united to perform a specific function.
Transcript	An object representation of a collection of *Exons*, that constitutes a probe in an Affymetrix Exon Array. It can be used to retrieve the associated *Exons, ExonArrayReporter*s and *Genes*.
TranscriptArrayReporter	An abstract superclass for *ExpressionArrayReporter* and *ExonReporter.*
Vocabulary	A collection of terms associated with a given Anomaly.

Table 6-1 caBIO domain objects and descriptions

Provenance Objects	***Description***
Provenance	An object modeling the provenance of various entities such as SNP, Gene, NucleicAcidSequence and Protein data in caBIO. It exposes information such as the source of the data (dbSNP, Unigene, LocusLink, Uniprot, etc) along with the class name modeling that object in caBIO, evidence code, etc.
InternetSource	An object modeling details of the primary data source providers for *SNP*, *Gene*, etc in caBIO
PublicationSource	An object modeling details of publications associated with caBIO data such as the author, date of publication, etc.
ResearchInstitutionSource	An object modeling details of the research institution such as name, address, contact information, etc from where caBIO data is obtained.
Source	An object modeling the primary data source for objects like SNP, Gene, Protein, etc in caBIO
SourceReference	An object providing web-enabled access to further details regarding Genes, Proteins, SNPs, and other caBIO objects from their primary data sources.
URLSourceReference	An object providing web-enabled access to obtain further details regarding Genes, Proteins, SNPs, and other caBIO objects from their primary data sources.

Table 6.2 Provenance objects and descriptions

Data Sources in the caBIO Database

This section describes the internal and external data sources for caBIO and how the information these sources provide can be accessed via caBIO objects.

The caBIO application programming interfaces were developed primarily in response to the need for programmatic access to the information at several NCI web sites, including:

- Cancer Genome Anatomy Project (CGAP)
- Cancer Therapy Evaluation Program (CTEP)
- CGAP Genetic Annotation Initiative (GAI)
- Cancer Molecular Analysis Project (CMAP)
- Affymetrix, Agilent and Illumina MicroArrays
- Genome Sequencing Center at University of California, Santa Cruz (UCSC)
- SNP Annotations from NCBI

- Protein Data from Uniprot-Swissprot
- SNP Annotations from SNP Consortium
- and others

While this information is, in theory, available from multiple public sites, the number of links to traverse and the extent of collation that would have to be performed is daunting. The CGAP, CMAP, and GAI web sites have distilled this information from both internal and public databases, and the caBIO data warehouses have optimized it for access with respect to the types of queries defined in the APIs.

While the caBIO data are extracted from many sources that include information from a wide variety of species, we emphasize that only genomic data pertaining to human and mouse are available from caBIO.

caBIO provides access to curated data from both internal (NCI) and external sites. Table 6.3 contains data source internal to NCI and Table 6.4 contains data sources from sites outside of NCI.

NCI Data Source	***Description***
CGAP	CGAP (Cancer Genome Anatomy Project) provides a collection of gene expression profiles of normal, pre-cancer, and cancer cells taken from various tissues. The CGAP interface allows users to browse these profiles by various search criteria, including histology type, tissue type, library protocol, and sample preparation methods. The goal at NCI is to exploit such expression profile information for the advancement of improved detection, diagnosis, and treatment for the cancer patient. Researchers have access to all CGAP data and biological resources for human and mouse, including ESTs, gene expression patterns, SNPs, cluster assemblies, and cytogenetic information. The CGAP web site provides a powerful set of interactive data-mining tools to explore these data, and the caBIO project was initially conceived as a programmatic interface to these tools and data. Accordingly, most of the data that are available from CGAP can also be accessed through the caBIO objects. Exceptions are those data sets having proprietary restrictions, such as the Mitleman Chromosome Aberration database. CGAP also provides access to lists of sequence-verified human and mouse cDNA IMAGE clones supplied by Invitrogen.
CMAP	CMAP (Cancer Molecular Analysis Project) is powered by caCORE. The goal of CMAP is to enable researchers to identify and evaluate molecular targets in cancer. The CMAP Profile Query tool finds genes with the highest or lowest expression levels (using SAGE and microarray data) for a given tissue and histology. Selecting a gene from the resulting table then leads to a Gene Info page. This page provides information about cytogenetic location, chromosome aberrations, protein similarities, curated and computed orthologs, and sequence-verified as well as full-length MGC clones, along with links to various other databases.

NCI Data Source	Description
CTEP	CTEP (Cancer Therapy Evaluation Program) funds an extensive national program of basic and clinical research to evaluate new anti-cancer agents, with a particular emphasis on translational research to elucidate molecular targets and drug mechanisms. In response to this emergent need for translational research, there has been a groundswell of translational support tools defining controlled vocabularies and registered terminologies to enhance electronic data exchange in areas that have heretofore been relatively non-computational. The caCORE trials data are updated with new CTEP data on a quarterly basis, and many of the objects are designed to support translational research. For example, a caCORE *Target* object represents a molecule of special diagnostic or therapeutic interest for cancer research, and an Anomaly object is an observed deviation in the structure or expression of a *Target*. An *Agent* is a drug or other intervention that is effective in the presence of one or more specific Targets. The *ClinicalTrialProtocol* object organizes administrative information pertaining to that protocol. Data from CTEP are used to populate the *Protocols, ProtocolAgents* and *ProtocolDiseases* objects.
GAI	GAI (CGAP Genetic Annotation Initiative) is an NCI research program to explore and apply technology for identification and characterization of genetic variation in genes important in cancer. The GAI utilizes data-mining to identify "candidate" variation sites from publicly available DNA sequences, as well as laboratory methods to search for variations in cancer-related genes. All GAI candidate, validated, and confirmed genetic variants are available directly from the GAI web site, and all validated SNPs have been submitted to the NCBI dbSNP database as well that are in turn available through the *SNP* .
HomoloGene	HomoloGene is an NCBI resource for curated and calculated gene homologs. The caBIO data sources capture only the calculated homologs stored by HomoloGene. These calculated homologs are the result of nucleotide sequence comparisons performed between each pair of organisms represented in UniGene clusters. caBIO provides this information via *HomologousAssociation* and updates this data on a monthly basis.
LocusLink/Entrez Gene	LocusLink contains curated sequence and descriptive information associated with a gene. Each entry includes information about the gene's nomenclature, aliases, sequence accession numbers, phenotypes, UniGene cluster IDs, OMIM IDs, gene homologies, associated diseases, map locations, and a list of related terms in the Gene Ontology Consortium's ontology. Sequence accessions include a subset of GenBank accessions for a locus, as well as the NCBI Reference Sequence. This information is available in *DatabaseCrossReference, GeneOntology, Clone*, etc amongst other objects.

NCI Data Source	***Description***
dbSNP	In collaboration with the National Human Genome Research Institute, the NCBI has established the dbSNP database to serve as a central repository for both single base nucleotide substitutions and short deletion and insertion polymorphisms. Once discovered, these polymorphisms could be used by additional laboratories, using the sequence information around the polymorphism and the specific experimental conditions. (Note that dbSNP takes the looser 'variation' definition for SNPs, so there is no requirement or assumption about minimum allele frequency.) The data from dbSNP is updated approximately every 3-4 months. Relevant information is available through *SNP, Provenance, Source, URLSourceReference, SourceReference, PhysicalLocation* and *Location* objects.
UniGene	Unigene provides a nonredundant partitioning of the genetic sequences contained in GenBank into gene clusters. Each cluster has a unique UniGene ID and a list of the mRNA and EST sequences that are included in that cluster. Related information stored with the cluster includes tissue types in which the gene has been expressed, mapping information, and the associated LocusLink, OMIM, and HomoloGene IDs, thus providing access to related information in those NCBI databases as well through *DatabaseCrossReference*. Because the information in UniGene is centered around genes, access to Unigene is provided via the caBIO *Gene* objects. Specifically, the method *getClusterId* associated with a *Gene* object can be used to fetch the gene's UniGene ID. The database IDs for the NCBI OMIM, LocusLink, Enzyme Commission and Ensembl databases can be obtained from *DatabaseCrossReference* using the *getDatabaseCrossReferenceCollection* method. While there is no explicit caBIO object corresponding to a Unigene cluster, all of the information associated with the cluster is available directly via the caBIO *Gene* object's methods. The sequences are available via *NucleicAcidSequence* and the associated aliases are available in the *GeneAlias*. Corresponding clone library information is exposed via *Clone* and *CloneRelativeLocation*. Appropriate 'provenance' information is available in *Provenance, Source, URLSourceReference* and *SourceReference* objects. Unigene data is updated on a monthly basis.

Table 6-2 NCI-related data sources in the caBIO database

External Data source	***Description***
Affymetrix	Affymetrix provides the majority of Microarray data for caBIO. The data provides information on allele frequencies of the SNP in different populations, and is represented by the *PopulationFrequency* object. The probeset information is available through *ExpressionArrayReporter* and *SNPArrayReporter* objects, amongst others. Details regarding the arrays exposed in caBIO are available through *Microarray* The *GeneRelativeLocation* object provides the location (intron, upstream, downstream etc.) of a SNP with respect to its associated genes. The validation status for a SNP comes from NCBI. The SNP Consortium (TSC) Ltd. a non-profit foundation provides the TSC id's for SNPs in *DatabaseCrossReference*.
Agilent	Data from Agilent's Whole Human Genome 44K Arrays and aCGH 244K Arrays are exposed through *MicroArray* and *ExpressionArrayReporter*.
BioCarta	BioCarta and its Proteomic Pathway Project (P3) provide detailed graphical renderings of pathway information concerning adhesion, apoptosis, cell activation, cell signaling, cell cycle regulation, cytokines/chemokines, developmental biology, hematopoeisis, immunology, metabolism, and neuroscience. NCI's CMAP web site captures pathway information from BioCarta, and transforms the downloaded image data into Scalable Vector Graphics (SVG) representations that support interactive manipulation of the online images. The CMAP web site displays BioCarta pathways selected by the user and provides options for highlighting *anomalies*, which include under- or over expressed genes as well as mutations. The pathway information is available via the *Pathway* object in caBIO. caBIO provides a class for manipulating SVG diagrams, which is described in SVG Manipulation Utility on page 36.
Illumina	One of the SNP-Arrays in caBIO. Data from Agilent is used to populate *MicroArray* and *ExpressionArrayReporter*, amongst others.
UniProt PIR	Universal Protein Resource (UniProt) is a complete annotated protein sequence database and is a central repository of protein sequence and function created by joining the information contained in Swiss-Prot, TrEMBL, and PIR. The UniProt Knowledge base provides access to extensive curated protein information, including the amino acid sequence, protein name or description, taxonomic data and protein aliases. caBIO exposes information from the Swissprot databanks through *ProteinSequence* and *ProteinAlias* objects. Mappings to RefSeq Ids are available through *DatabaseCrossReference.* Provenance-related information is available through *Provenance, Source, SourceReference* and *URLSourceReference* objects. Protein Domain information from Interpro is exposed through *ProteinDomain* object

External Data source	Description
Gene Ontology Consortium	The Gene Ontology Consortium provides a controlled vocabulary for the description of molecular functions, biological processes, and cellular components of gene products. The terms provided by the consortium define the recognized attributes of gene products and facilitate uniform queries across collaborating databases. In general, each gene is associated with one or more biological processes, and each of these processes may in turn be associated with many genes. In addition, the GO ontologies define many parent/child relationships among terms. For example, a branch of the ontology tree under biological process contains the term "cell cycle control", which in turn bifurcates into the "child" terms cell cycle arrest, cell cycle checkpoint, control of mitosis, etc. caBIO does not extract ontology terms directly from the Gene Ontology Consortium but, instead, extracts those terms stored with the LocusLink entry for that gene. This information is available via *GoClosure, GoGenes*, *GeneOntology* and *GeneOntologyRelationship* objects.
SNP Consortium	The SNP Consortium Ltd. is a non-profit foundation organized for providing public genomic data. Its mission is to develop up to 300,000 SNPs distributed evenly throughout the human genome and to make the information related to these SNPs available to the public without intellectual property restrictions. The TSC Ids corresponding to a SNP are available through *DatabaseCrossReference.*
UCSC	UCSC (University of California, Santa Cruz Distributed Annotation System) provides the data for the Chromosomal start and end positions of mRNA and EST sequences. The positions of cytogenetic bands within a chromosome, represented by the caBIO `Cytoband` object, are also obtained from the UCSC. This data is used to populate the *PhysicalLocation*, *Cytoband, CytogeneticLocation* and *Location* objects with the locations of ESTs, MRNAs and Cytobands respectively. The ESTs and MRNAs in *PhysicalLocation* and *Location* are linked with the corresponding sequence identifiers from *NucleicAcidSequence*

Table 6-3 External data sources in the caBIO database

caGrid Identifiers

The functionality provided by caGrid's Identifier Services Framework and the integration in caBIO is to have "identifiers" for each caBIO domain object. The identifier is essentially a forever globally unique name for the caBIO data-object such that it can be unambiguously used to refer to the data from different application contexts.

This identifier, obtained from the caGrid Identifier Service Framework, is essentially a string and a forever globally unique name for the caBIO domain object. Furthermore, the identifier can be (globally) resolved to the associated caBIO domain object.

In order to abstract the identifier's object properties, the data service implementations and the resolution mechanisms, the identifier's value must be treated as a "meaningless" opaque string by the consumer applications. Any leaking of design and implementation choices for the identifier framework in the applications is undesirable from an architecture point of view as it makes the implementations brittle and susceptible to future changes. Of course resolution information will have to be embedded in identifier name, but this should only be meaningful for resolution service related components that are layered below the application.

caBIO Specific Utilities

Manipulating SVG Diagrams

caBIO provides a utility class called SVGManipulator for manipulating pathway SVG diagrams. BioCarta and its Proteomic Pathway Project (P3) provide detailed graphical renderings of pathway information. NCI's CMAP web site captures pathway information from BioCarta, and transforms the downloaded image data into Scalable Vector Graphics (SVG) representations that support interactive manipulation of the online images. The SVGManipulator utility class provides the capability to do the following:

- Change the display colors for each gene contained in an SVG diagram.
- Modify the URL linking a gene in the SVG diagram to external gene information. The default gene URL links to the CMAP website.
- Disable all genes or nodes within an SVG diagram.
- Retrieve a gene's color.
- Reset a gene or node to its original state.
- Retrieve/set SVG diagram attributes via getter/setter methods.

Figure 6-1 shows an example of how to use *SVGManipulator* class to modify content of an SVG diagram. This example uses the *gov.nih.nci.cabio.domain.Pathway* and *gov.nih.nci.cabio.domain.Gene* objects.

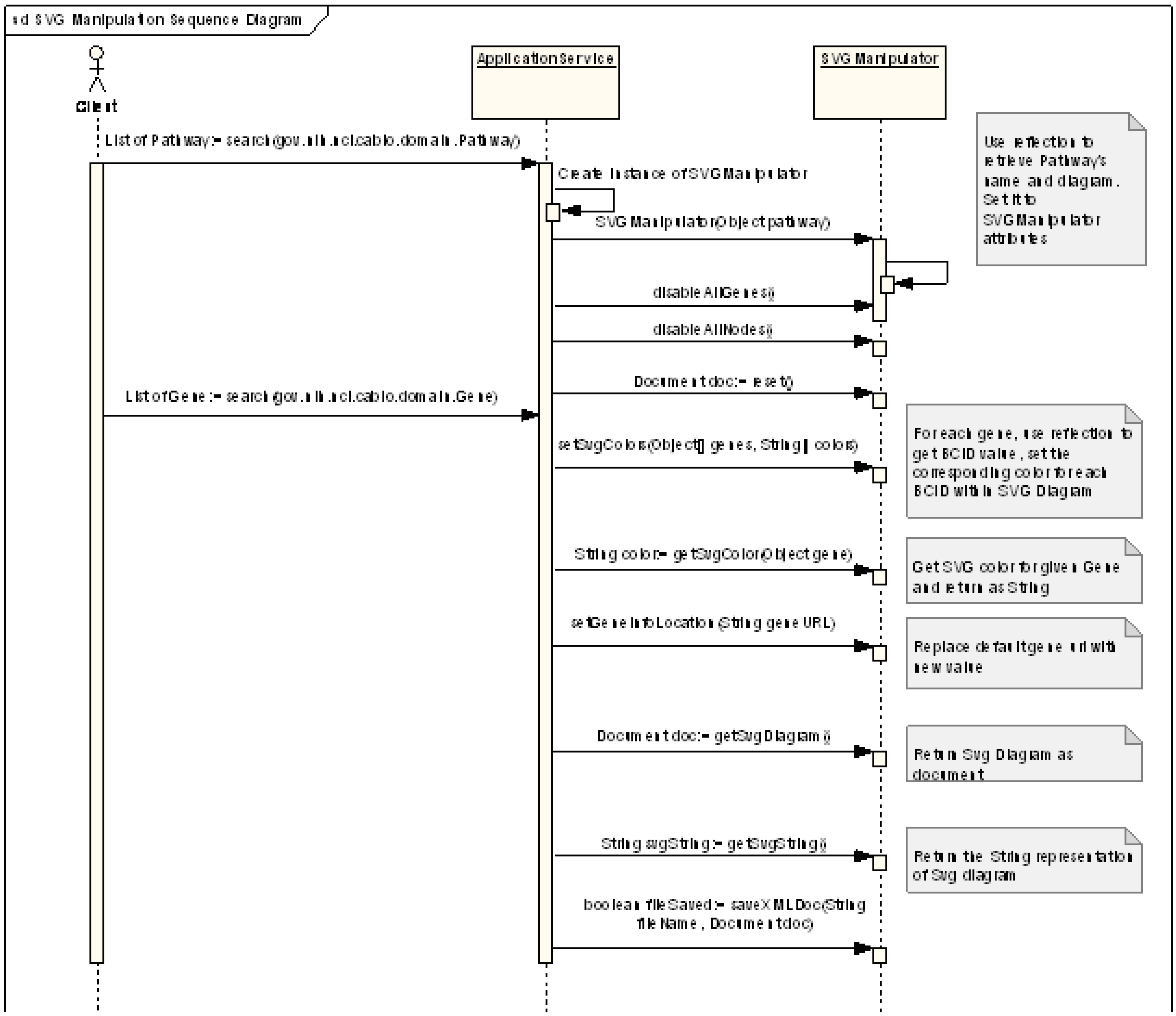


Figure 6-1 Sequence diagram using caBIO Pathway and Gene objects

SVG Diagram Manipulation Utility Example

The following test client demonstrates how to use the *SVGManipulation* class to change the appearance of a pathway diagram associated with a given pathway.

```
import gov.nih.nci.system.applicationservice.*;
import java.util.*;

import gov.nih.nci.camod.domain.*;
import gov.nih.nci.camod.domain.impl.*;
```

```
import gov.nih.nci.cadsr.domain.*;
import gov.nih.nci.cadsr.domain.impl.*;

import gov.nih.nci.cabio.domain.*;
import gov.nih.nci.cabio.domain.impl.*;
import gov.nih.nci.common.util.*;

import org.hibernate.criterion.*;

public class TestClient {


    public static void main(String[] args) {

       System.out.println("*** TestClient...");
    try{


ApplicationService appService =
ApplicationServiceProvider.getApplicationService();

    /**** Test scenarios for SVG Pathway Diagram using
SVGManipulator.java *****/

    try {

        System.out.println("Using basic search.  Retrieving Pathway");
        Pathway pw = new PathwayImpl();
        pw.setId(new Long(251));

        try {
            List resultList = appService.search(Pathway.class, pw);
            System.out.println("result count: " + resultList.size());
            for (Iterator resultsIterator = resultList.iterator();
                    resultsIterator.hasNext();) {
                Pathway returnedPw = (Pathway) resultsIterator.next();
                String pathwayDiagram = returnedPw.getDiagram();

                SVGManipulator svgM = new SVGManipulator(returnedPw);
                Document orgSvgDoc = svgM.getSvgDiagram();

                Document org0 = svgM.reset();

                svgM.disableAllGenes();
                Document disableGenesDoc = svgM.getSvgDiagram();
                disableGenesDoc);

                Document org1 = svgM.reset();

                svgM.disableAllNodes();
```

```
Document disableNodesDoc = svgM.getSvgDiagram();
disableNodesDoc);

Document org = svgM.reset();

Gene[] genes= new Gene[2];
String[] colors=new String[2];

Gene p53=new GeneImpl();
p53.setId(new Long(1031));
List resultList1 = appService.search(Gene.class, p53);
if(resultList1.size()> 0)
    genes[0]=(Gene)resultList1.get(0);

Gene p54=new GeneImpl();
p54.setId(new Long(2));
List resultList2 = appService.search(Gene.class, p54);
genes[1] = (Gene)resultList2.get(0);
colors[0]="255,255,255";
colors[1]="0,255,255";

svgM.setSvgColors(genes, colors);

Document geneColors = svgM.getSvgDiagram();

String genep53color = svgM.getSvgColor(genes[0]);
System.out.println("geneP53 color: " + genep53color);

Document org10 = svgM.reset();

String genep53color1 = svgM.getSvgColor(genes[0]);
System.out.println("geneP53 color1: " + genep53color1);
String svgString = svgM.toString();
System.out.println("toString:\n" + svgString);

svgM.setGeneInfoLocation("http://www.google.com");

Document geneLocation = svgM.getSvgDiagram();

// Using Map
Map geneColors = new HashMap();
geneColors.put("rab7", "0,255,255");
geneColors.put("rab1", "0,255, 255");

svgM.setSvgColors(geneColors);

Document d = svgM.getSvgDiagram();
```

```
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        } catch (RuntimeException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
          }
        }
    }
```

Appendix A References

Technical Manuals/Articles

1. National Cancer Institute. *caCORE SDK 4.0 Developer's Guide* http://ncicb.nci.nih.gov/NCICB/infrastructure/cacoresdk#Documentation
2. Java Bean Specification: http://java.sun.com/products/javabeans/docs/spec.html
3. Foundations of Object-Relational Mapping: http://www.chimu.com/publications/objectRelational/
4. Object-Relational Mapping articles and products: http://www.service-architecture.com/object-relational-mapping/
5. Hibernate Reference Documentation: http://www.hibernate.org/5.html
6. Basic O/R Mapping: http://www.hibernate.org/hib_docs/v3/reference/en/html/mapping.html
7. Java Programming: http://java.sun.com/learning/new2java/index.html
8. Jalopy User Manual: http://jalopy.sourceforge.net/existing/manual.html
9. Javadoc tool: http://java.sun.com/j2se/javadoc/
10. JUnit: http://junit.sourceforge.net/
11. Extensible Markup Language: http://www.w3.org/TR/REC-xml/
12. XML Metadata Interchange: http://www.omg.org/technology/documents/formal/xmi.htm
13. Ehcache: http://ehcache.sourceforge.net/documentation/

Scientific Publications

1. Ansher SS and Scharf R (2001). The Cancer Therapy Evaluation Program (CTEP) at the National Cancer Institute: industry collaborations in new agent development. Ann N Y Acad Sci 949:333-40.
2. Boon K, Osorio EC, Greenhut SF, Schaefer CF, Shoemaker J, Polyak K, Morin PJ, Buetow KH, Strausberg RL, De Souza SJ, and Riggins GJ (2002). An anatomy of normal and malignant gene expression. Proc Natl Acad Sci U S A 2002 Jul 15.
3. Buetow KH, Klausner RD, Fine H, Kaplan R, Singer DS, and Strausberg RL (2002). Cancer Molecular Analysis Project: Weaving a rich cancer research tapestry. Cancer Cell 1(4):315-8.
4. Boguski & Schuler (1995). ESTablishing a human transcript map. Nature Genetics 10: 369-71.

5. Clifford R, Edmonson M, Hu Y, Nguyen C, Scherpbier T, and Buetow KH (2000). Expression-based genetic/physical maps of single-nucleotide polymorphisms identified by the Cancer Genome Anatomy Project. Genome Res 10(8):1259-65.
6. Covitz P.A., Hartel F., Schaefer C., De Coronado S., Sahni H., Gustafson S., Buetow K. H. (2003). caCORE: A common infrastructure for cancer informatics. Bioinformatics. *19*: 2404-2412.
7. Dowell RD, Jokerst RM, Day A, Eddy SR, Stein L. The Distributed Annotation System. BMC Bioinformatics 2(1):7.
8. The Gene Ontology Consortium. (2000). Gene ontology: tool for the unification of biology. Nature Genetics 25:25-9.
9. The Gene Ontology Consortium. (2001). Creating the gene ontology resource: design and implementation. Genome Res 11:1425-33.
10. Hartel FW and de Coronado S (2002). Information standards within NCI. In: *Cancer Informatics: Essential Technologies for Clinical Trials*. Silva JS, Ball MJ, Chute CG, Douglas JV, Langlotz C, Niland J and Scherlis W, eds. Springer-Verlag.
11. Pruitt KD, Katz KS, Sicotte H, and Maglott DR (2000). Introducing RefSeq and LocusLink: curated human genome resources at the NCBI. Trends Genet 16(1):44-7.
12. Pruitt KD, and Maglott DR (2001). RefSeq and LocusLink: NCBI gene-centered resources. Nucleic Acids Res 29(1):137-40.
13. Schuler et al. (1996). A gene map of the human genome. Science 274: 540-6.
14. Schuler (1997). Pieces of the puzzle: expressed sequence tags and the catalog of human genes. J Mol Med 75(10):694-8.
15. Strausberg RL (1999). The Cancer Genome Anatomy Project: building a new information and technology platform for cancer research. In: Molecular Pathology of Early Cancer (Srivastava S, Henson DE, Gazdar A, eds. IOS Press, 365-70.
16. Strausberg RL (2001). The Cancer Genome Anatomy Project: new resources for reading the molecular signatures of cancer. J Pathol 195:31-40.
17. Zhang, Schwartz, Wagner, and Miller (2000). A Greedy algorithm for aligning DNA sequences. J Comp Biol 7(1-2):203-14.
18. Sherry ST, Ward MH, Kholodov M, Baker J, Phan L, Smigielski EM, Sirotkin K (2001). dbSNP: the NCBI database of genetic variation. Nucleic Acids Res. 1;29(1):308-11.
19. Karolchik D, Baertsch R, Diekhans M, Furey TS, Hinrichs A, Lu YT, Roskin KM, Schwartz M, Sugnet CW, Thomas DJ, Weber RJ, Haussler D and Kent WJ(2003) The UCSC Genome Browser Database. Nucl. Acids Res 31(1), 51-54
20. Lennon G, Auffray C, Polymeropoulos M, Soares MB(1996) The I.M.A.G.E. Consortium: an integrated molecular analysis of genomes and their expression. Genomics. 33(1):151-2
21. Wheeler DL, Barrett T, Benson DA, Bryant SH, Canese K, Chetvernin V, Church DM, DiCuccio M, Edgar R, Federhen S, Geer LY, Kapustin Y, Khovayko O, Landsman D, Lipman DJ, Madden TL, Maglott DR, Ostell J, Miller V, Pruitt KD, Schuler GD, Sequeira E, Sherry ST, Sirotkin K, Souvorov A, Starchenko G, Tatusov RL, Tatusova TA, Wagner L, Yaschenko E.(2007) Epub(2006) Database resources of the National Center for Biotechnology Information. Nucleic Acids Res. 35(Database issue):D5-12

22. Pontius JU, Wagner L, Schuler GD (2003) UniGene: a unified view of the transcriptome In: The NCBI Handbook. Bethesda (MD): National Center for Biotechnology Information.

23. Schuler GD (1997) Pieces of the puzzle: expressed sequence tags and the catalog of human genes. J Mol Med 75:694-698

24. Schuler GD, et al (1996) A gene map of the human genome. Science 274:540-546.

25. Boguski MS, Schuler GD (1995) ESTablishing a human transcript map. Nature Genetics 10: 369-371.

26. Thorisson GA, Stein LD (2003) The SNP Consortium website: past, present and future. Nucleic Acids Res. 31(1):124-7.

caBIG Material

1. caBIG: http://cabig.nci.nih.gov/
2. caBIG Compatibility Guidelines: http://cabig.nci.nih.gov/guidelines_documentation

caCORE Material

1. caCORE: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview
2. caBIO: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/caBIO
3. caDSR: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/cadsr
4. EVS: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary
5. CSM: http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/csm

Modeling Concepts

1. Enterprise Architect Online Manual: http://www.sparxsystems.com.au/EAUserGuide/index.html
2. OMG Model Driven Architecture (MDA) Guide Version 1.0.1: http://www.omg.org/docs/omg/03-06-01.pdf
3. Object Management Group: http://www.omg.org/

Applications Currently Using caBIO

1. BIO Browser: http://www.jonnywray.com/java/index.html
2. caPathway: http://cgap.nci.nih.gov/Pathways

Appendix B Migration Guide

Overview of Changes in 4.0

The caBIO API is based on the caCORE SDK, which has undergone dramatic refactoring in its 4.0 release. A separate migration guide exists for the SDK, but it is aimed at SDK developers and not all of it applies to caBIO. Additionally, caBIO itself has undergone changes in the 4.0 release. The purpose of this appendix is to provide a comprehensive migration guide for users of the caBIO API.

The following is an overview of the main changes in the SDK and caBIO API that will affect end users:

- The caBIO API was separated out from combined caCORE API
- Java API
    - The API used to obtain an ApplicationService instance has changed.
    - Some classes have moved across packages.
    - It is no longer possible to specify the number of records to fetch from the server.
    - The setSearchCaseSensitivity method has been removed.
    - Domain object Impl classes, deprecated in the last release, have now been removed.
    - JAR files in the client distribution have changed. New jars have been added, and some existing jars were upgraded.
- Web Services API
    - The .ws packages have been eliminated, along with all the web service beans. Normal domain objects are now to be used with the web service.
    - Web service methods have changed
- Domain Objects
    - Several attributes exposing internal foreign keys have been removed from the CytogeneticLocation and Provenance classes.
    - The association cardinality between Cytoband and PhysicalLocation has been reversed.
    - Associations to GeneRelativeLocation have been replaced with links to its new super class RelativeLocation.

Each of these areas is discussed in detail in the following sections.

Note: This migration guide is not intended to discuss or list the new caBIO 4.0 features. For information related to the new caBIO 4.0 features, see New Features in caBIO 4.0 on page 1.

Changes Required When Migrating to caBIO 4.0

Separation from the caCORE API

The caCORE API has been split into three separate APIs for caBIO, caDSR and EVS. This guide only addresses the caBIO API. However, it is important to note that caCORE users, who depend on classes in more than one of these projects, will now need to install and use multiple APIs.

Java API Changes

Changes to ApplicationService API

To obtain an ApplicationService instance, users must use the ApplicationServiceProvider introduced in version 3.2. To obtain the default service:

```
ApplicationService appService =
    ApplicationServiceProvider.getApplicationService();
```

To obtain an ApplicationService for a specific URL, the API has changed as follows:

3.2	`ApplicationService appService = ApplicationService.getRemoteInstance(url);`
4.0	`ApplicationService appService = ApplicationServiceProvider.getApplicationServiceFromUrl(url),`

Furthermore, caBIO-specific functionality has been split into a subclass called CaBioApplicationService. This functionality includes all Grid Id and FreestyleLM methods. To get access to this functionality, cast the object like this:

```
CaBioApplicationService appService = (CaBioApplicationService)
    ApplicationServiceProvider.getApplicationServiceFromUrl(url);
```

Some classes have moved across packages

The *ApplicationServiceProvider* class has been moved from gov.nih.nci.system.applicationservice to gov.nih.nci.system.client.

The *HQLCriteria* class has moved from gov.nih.nci.common.util to gov.nih.nci.system.query.hibernate.

Users of these classes should change the import statements accordingly.

No longer possible to specify the number of records to fetch

The SDK 4.0 (and thus caBIO) no longer supports custom record fetch sizes.

No longer possible to specify search case sensitivity

The SDK no longer provides a setSearchCaseSensitivity() method on the ApplicationService class. In previous versions, this method did not have any effect, so the behavior remains consistent. The case sensitivity defaults to false meaning that all queries all case-insensitive.

Deprecated Impl classes have been removed

Users of caBIO 3.2 could instantiate domain objects using the deprecated Impl classes in the impl packages. These classes have been eliminated in 4.0 so that there is now only one way to instantiate domain objects:

3.2	`Gene gene = new Gene();`
	`// or this`
	`Gene gene = new GeneImpl();`

4.0	`Gene gene = new Gene();`

JAR files in the client distribution have changed

New jars have been added to support FreestyleLM search. Certain existing jars, such as Hibernate, were upgraded. Consequently, users should pick up all the jars from the 4.0 distribution when using the 4.0 API.

Web Services API

The .ws packages has been eliminated

The web services are now based on the regular domain objects, and the .ws packages have been removed. Users of the web services should update their import statements accordingly.

Domain Objects

Foreign Key Attributes have been removed

Several attributes exposing internal foreign keys have been removed from the model. The data is still available at the associated object.

3.2	4.0
`Provenance.getImmediateSourceId()`	`Provenance.getImmediateSource().getId()`
`Provenance.getOriginalSourceId()`	`Provenance.getOriginalSource().getId()`
`Provenance.getSupplyingSourceId()`	`Provenance.getSupplyingSource().getId()`
`CytogeneticLocation.getStartCytobandLocId()`	`CytogeneticLocation.getStartCytoband().getId()`
`CytogeneticLocation.getEndCytobandLocId()`	`CytogeneticLocation.getEndCytoband().getId()`

Reversed association between Cytoband and PhysicalLocation

In the previous release, the cardinality of the relationship between these two objects was one PhysicalLocation to many Cytobands. This has been corrected so that one Cytoband can now be associated with many PhysicalLocations. The underlying data only exhibits a 1-to-1 relationship for the time being.

3.2	`cytoband.getPhysicalLocation();` `physicalLocation.getCytobandCollection();`
4.0	`cytoband.getPhysicalLocationCollection();` `physicalLocation.getCytoband();`

Users who call these methods will need to modify their code accordingly.

Associations to GeneRelativeLocation replaced with RelativeLocation

The getGeneRelativeLocationCollection method in the SNP and Gene classes no longer exists. A new method, getRelativeLocationCollection, retrieves all associated subclasses of RelativeLocation including GeneRelativeLocation. Users looking for only GeneRelativeLocations will need to filter the result set.

Glossary

The following table contains a list of terms used in this document, with accompanying definitions.

Term	*Definition*
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
Application Service	This refers to the CSM interface which exposes all the writeable as well as business methods for a particular application
Axis	Open source package from Apache that provides SOAP-based web services to users
caBIG	cancer Biomedical Informatics Grid
caBIO	Cancer Bioinformatics Infrastructure Objects
caCORE	Cancer Common Ontologic Representation Environment
caDSR	Cancer Data Standards Repository
caMOD	Cancer Models Database
Cardinality	Cardinality describes the minimum and maximum number of associated objects within a set
CGAP	Cancer Genome Anatomy Project
CLM	Common Logging Module
CMAP	Cancer Molecular Analysis Project
CQL	caBIG Query Language
CSM	Common Security Module
CTEP	Cancer Therapy Evaluation Program
DAO	Data Access Objects
EA	Enterprise Architect

Term	*Definition*
EBI	European Bioinformatics Institute
EMF	Eclipse Modeling Framework
EVS	Enterprise Vocabulary Services
FreeMarker	A "template engine"; a generic tool to generate text output (anything from HTML or RTF to auto generated source code) based on templates.
FreeStyleLM	Freestyle Lexical Mine search
GAI	CGAP Genetic Annotation Initiative
GEDP	Gene Expression Data Portal
Hibernate	A high performance object/relational persistence and query service for JavaProvides the ability to develop persistent classes following common object-oriented (OO) design methodologies such as association, inheritance, polymorphism, and composition (http://www.hibernate.org)
HQL	Hibernate Query Language is designed as a "minimal" object-oriented extension to SQL, provides a bridge between the object and relational databases
IDE	Integrated Development Environment
ISO	International Organization for Standardization
JAR	Java Archive
Java Bean	Reusable software components that work with Java
Java Servlet	Server-side Java programs, that web servers can run to generate content in response to client requests
Javadoc	Tool for generating API documentation in HTML format from doc comments in source code (http://java.sun.com/j2se/javadoc/)
JBoss	J2SE application server used as a presentation layer in caCORE architecture. See also Tomcat.
JDBC	Java Database Connectivity

Term	*Definition*
JDiff	Javadoc doc-let which generates an HTML report of all the packages, classes, constructors, methods, and fields which have been removed, added or changed in any way, including their documentation, when two APIs are compared (http://javadiff.sourceforge.net/)
JET	Java Emitter Templates
JMI	Java Metadata Interface
JSP	Java Server Pages. Web pages with Java embedded in the HTML to incorporate dynamic content in the page
JUnit	A simple framework to write repeatable tests (http://junit.sourceforge.net/)
Lucene	Open source package from Apache that provides a full-text search engine
MDA	Model driven architecture.
Metadata	Definitional data that provides information about or documentation of other data.
MMHCC	Mouse Models of Human Cancers Consortium
NCI	National Cancer Institute
NCICB	National Cancer Institute Center for Bioinformatics
NSC	Nomenclature Standards Committee
OMG	Object Management Group
ORM	Object Relational Mapping
Persistence Layer	Data storage layer, usually in a relational database system
QBE	Query By Example
RDBMS	Relational Database Management System
REST	Representational State Transfer

Term	*Definition*
SOAP	Simple Object Access Protocol. A lightweight XML-based protocol for the exchange of information in a decentralized, distributed environment
SQL	Structured Query Language
SVG	Scalar Vector Graphic
Tomcat	J2SE application server used as a presentation layer in caCORE architecture. See also JBoss.
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locators
war	Web archive file
WSDL	Web Services Description Language
XMI	XML Metadata Interchange (http://www.omg.org/technology/documents/formal/xmi.htm) - The main purpose of XMI is to enable easy interchange of metadata between modeling tools (based on the OMG-UML) and metadata repositories (OMG-MOF) in distributed heterogeneous environments
XML	Extensible Markup Language (http://www.w3.org/TR/REC-xml/) - XML is a subset of Standard Generalized Markup Language (SGML). Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML
XSL/XSLT	Extensible stylesheet language for expressing stylesheets and XSL Transformations (XSLT).

Index