NCI PROTÉGÉ 1.3

Editor's Guide

Center for Bioinformatics

December 17, 2008

CONTENTS

Chapter 7
Working with Properties 99

Chapter 8
Splitting, Merging, and Cloning Classes 119

Chapter 9
Pre-Retiring and Retiring Classes 133

LIST OF FIGURES

ABOUT THIS GUIDE

This section introduces you to the *NCI Protégé 1.3 Editor's Guide*.

Topics in this Section

- *Purpose* on this page
- *Audience* on this page
- *Topics Covered* on page 2
- *Additional References* on page 3
- *Text Conventions Used* on page 3
- *Credits and Resources* on page 4
- *Application Support Contacts* on page 5

Purpose

This guide is a reference for editors using Protégé at the National Cancer Institute (NCI). It discusses the semantic model that supports the NCI Protégé application. It also explains how to install the application and how to use it to build and manage ontologies for the NCI Thesaurus.

Audience

Typical User

This guide is designed for the following users:

- Existing NCI ontology editors (modelers) who have used the TDE application and who are now using NCI Protégé.
- New NCI ontology editors who need background information on the semantic model used at the NCI, an understanding of the editing workflow, and knowledge of processes and procedures for using NCI Protégé.

Note: This guide is not intended for administrators, developers, or end users of the generated vocabulary.

Prerequisites

To get the most out of this guide, you should be familiar with the following topics:

- Core editing concepts
- General knowledge of ontology structures
- General familiarity with the Web Ontology Language (OWL) representations of ontologies.

Topics Covered

If you have worked with previous versions of NCI Protégé, see *Additional References* on this page.

If you are new to NCI Protégé, read the following overview to learn what you will find in each chapter and appendix.

- *Chapter 1* gives an overview of the Enterprise Vocabulary Services (EVS).
- *Chapter 2* gives an overview of description logics and how they are used in the semantic model for the NCI Thesaurus.
- *Chapter 3* explains where to find the NCI Protégé installation files, how to install the application, how to log in, and how to set up an optional subset of the NCI Thesaurus on your hard drive.
- *Chapter 4* provides an overview of the NCI Protégé interface.
- *Chapter 5* explains simple and advanced searching techniques and provides several query examples.
- *Chapter 6* provides an overview of classes in NCI Protégé and explains how to create and tree a class.
- *Chapter 7* explains how to use the Basic Data, Relations, and Properties subtabs to create and manage properties.
- *Chapter 8* explains how to split, merge, and clone classes.
- *Chapter 9* explains the processes and procedures for pre-retiring and retiring a class.
- *Chapter 10* introduces three useful Protégé utilities: the Report Writer, Batch Loader, and Batch Editor subtabs.
- *Chapter 11* provides an overview of the BiomedGT terminology and the workflow processes and procedures used by external collaborators and NCI terminology curators.
- *Appendix A* lists terms used in the Apelon Terminology Development Environment (TDE) and gives their equivalent terms in Protégé.

Additional References

For more information about OWL, the original Protégé application, and NCI Protégé, see the following references:

- The W3C OWL specification: http://www.w3.org/2004/OWL/
- Stanford Protégé home page: http://protege.stanford.edu/
- Access to software downloads and tutorials: http://www.co-ode.org/
- Documentation available on GForge: https://gforge.nci.nih.gov/docman/?group_id=174.

Text Conventions Used

This section explains conventions used in this guide. The various typefaces represent interface components, keyboard shortcuts, toolbar buttons, dialog box options, and text that you type.

Convention	***Description***	***Example***
Bold	Highlights names of interface elements such as option buttons, check boxes, drop-down menus, menu commands, command buttons, or icons.	Click **Search**.
URL	Indicates a Web address.	http://domain.com
Text in SMALL CAPS	Indicates a keyboard shortcut.	Press ENTER.
Text in SMALL CAPS + Text in SMALL CAPS	Indicates keys that are pressed simultaneously.	Press SHIFT + CTRL.
Italics	Highlights newly introduced terms; also highlights references to other documents, sections, figures, and tables.	This is known as *cloning*. See *Figure 4.5*.
`Italic monospaced type`	Represents text that you type.	In the **New Subset** text box, enter *`Proprietary Proteins`*.
Note:	Highlights information of particular importance	**Note:** This concept is used throughout the document.
{ }	Surrounds replaceable items.	Replace {last name, first name} with the Principal Investigator’s name.

Credits and Resources

The following people contributed to the development of this guide.

NCI Protégé Development and Management Teams		
Development	***Quality Assurance***	***Project and Product Management***
Robert Dionne[2]	Maria-Elena Hernandez[6]	Frank Hartel[1]
Timothy Redmond[5]	Steven Hunter[7]	Gilberto Fragoso[1]
Tania Tudorache[5]	Tracy Safran[3]	Sherri De Coronado[1]
Natasha Noy[5]		Laura Roth[2]
Kim Ong[4]		Carol Creech[2]
Iris Guo[4]		
Chris Callendar[6]		
Documentation	***Systems and Application Support***	***User Community Resources***
Edward VanArsdall[2]	Tracy Safran[3]	John Bradsher[2]
	Robert Wynne[2]	Liz Hahn-Dantona[2]
		Nicole Thomas[2]
[1] National Cancer Institute Center for Bioinformatics (NCICB)	[2] Lockheed Martin	[3] Science Application International Corporation (SAIC)
[4] Northrup Grumman	[5] Stanford Medical Informatics (SMI)	[6] University of Victoria
[7] Ekagra Software Technologies		

Application Support Contacts

To obtain general information about NCI Protégé, receive support, or report a bug, contact NCICB Application Support.

Support Option	***Additional Information***
For NCI Protégé Users	
GForge	If you are an NCI Protégé user, post all bug reports and requests for enhancements on GForge using the following URL: https://gforge.nci.nih.gov/tracker/?group_id=174 For instructions on reporting bugs, see *Submitting a Bug Report and File Attachment to GForge* on page 48.
Administrative tasks for NCI Protégé	EVS Operations Support can provide the following services: ● Loading a database ● Creating a user account ● Creating or configuring a Protégé project ● Installing or reinstalling the Protégé client on a user's machine **Note:** If you need any of these services, send your request to the e-mail address under the **For General Users** section (below).
For General Users	
E-mail	Write to ncicb@pop.nci.nih.gov and include the following information: ● Your contact information, including your phone number ● The name of the application that you are using ● The URL (for Web-based applications) ● A description of the problem and the steps required to recreate it ● The text of any error messages you have received.
Web	NCICB support: http://ncicbsupport.nci.nih.gov/sw/ NCI Protégé online support forums: https://gforge.nci.nih.gov/forum/?group_id=174
Telephone	Local: 301-451-4384 Toll-free: 888-478-4423 Telephone support is available Monday through Friday, 8 a.m. - 8 p.m. Eastern time, excluding government holidays.

CHAPTER
1
ABOUT THE ENTERPRISE VOCABULARY SERVICES

This chapter introduces you to the NCI Enterprise Vocabulary Services (EVS) and its terminologies.

Topics in this Chapter

Introduction

The NCI Enterprise Vocabulary Services (EVS) is a partnership between the NCI Center for Bioinformatics and the NCI Office of Communications. Since 1997, EVS has worked to harmonize and integrate the many diverse terminologies and coding frameworks used by the NCI and its partners.

EVS serves a critical need by providing a well-designed ontology covering cancer science. Such an ontology is required for data annotation, inferencing, and other functions. Annotated data range from genomic sequences and case report forms to cancer image data.

EVS is active in NIH-wide harmonization initiatives, federal standards development efforts, and other standards development organizations. These activities help to develop terminology resources and software tools to facilitate compatible coding, retrieval, and aggregation of biomedical information.

Key EVS Terminologies

The establishment of controlled vocabularies is important to any application involving electronic data sharing. The importance of controlled vocabularies is perhaps most apparent in clinical trials data collection and data reporting. It is also important in general data annotation of any kind.

To respond to the need for consistency among various NCI projects and initiatives, the NCI publishes the NCI Thesaurus, the NCI Metathesaurus, and BiomedGT. The caCORE EVS interfaces, discussed later in this guide, provide access to these terminologies.

NCI Thesaurus

EVS publishes the NCI Thesaurus as a core reference terminology and biomedical ontology. Implemented as a Description Logic vocabulary, the Thesaurus is a self-contained, logically consistent terminology providing rich textual and ontological descriptions of some 50,000 key biomedical concepts.

EVS developed the NCI Thesaurus in response to a need for consistent shared vocabularies among various projects and initiatives at the NCI and in the entire cancer research community. Published monthly, the Thesaurus is used in a growing number of NCI and other systems.

NCI Metathesaurus

The NCI Metathesaurus is a comprehensive biomedical terminology database that houses 1,100,000 concepts mapped to 2,500,000 terms with 5,000,000 relationships. Based on the National Library of Medicine’s Unified Medical Language System Metathesaurus (UMLS), the NCI Metathesaurus includes most UMLS terms and supplements them with additional cancer-centric vocabulary. The Metathesaurus excludes certain proprietary vocabularies and includes others with restricted use.

The NCI developed some of the NCI Metathesaurus vocabularies and licensed others. *Table 1.1* describes vocabularies that were locally developed. Note that a limited model of the NCI Thesaurus is accessible through the Metathesaurus as the NCI Source. Additional external, proprietary vocabularies include MedDRA, SNOMED, and ICD-O-3, among others.

Vocabulary	***Content***	***Usage***
NCI Source	Limited model of the NCI Thesaurus	Reference terminology for cancer research applications
NCIPDQ	Expanded and re-organized PDQ	CancerLit indexing and clinical trials accrual
NCISEER	SEER terminology	Incidence reporting
CTEP	CTEP terminology	Clinical trials administration
MDBCAC	Topology and morphology	Cancer genome research
ELC2001	NCBI tissue taxonomy	Tissue classification for genetic data such as cDNA libraries.

Table 1.1 NCI local source vocabularies included in the Metathesaurus

Vocabulary	*Content*	*Usage*
ICD03	Oncology classifications	Cancer genome research and incidence reporting
MedDRA	Regulatory reporting terminology	Adverse event reporting
MMHCC	Mouse Cancer Database terminology	Mouse Models of Human Cancer Consortium
CTRM	Core anatomy, diagnosis, and agent terminology	Translational research by NCICB applications

Table 1.1 NCI local source vocabularies included in the Metathesaurus (Continued)

Unlike the NCI Thesaurus, the NCI Metathesaurus is not designed to provide unequivocal or consistent definitions. Like the UMLS, its purpose is to provide mappings of terms across vocabularies.

BiomedGT

BiomedGT is an open, collaboratively developed terminology for translational research. This terminology has a wiki-based workflow component. For detailed coverage of BiomedGT, see *Workflow for NCI Editors* on page 161.

EVS APIs and Server Hosting

The EVS terminology service is a component of caCORE. The EVS/caCORE 4.x service currently supports two APIs. This topic describes both APIs and the server hosting environment for each.

caCORE 3.2 EVS API

The caCORE 3.2 EVS API provides access to two server components: (1) the NCI Metaphrase server, which hosts the Metathesaurus database; and (2) the NCI Distributed Terminology Server (DTS), which hosts the NCI Thesaurus and several other vocabularies.

The NCI licenses the Metaphrase and DTS servers from Apelon, Inc. Both servers have proprietary Java APIs that are not available to the public. The NCI has extended and otherwise modified the Metaphrase and DTS servers to provide functionality that is not available in the commercial version of these products. To expose the functionality, we developed a public domain open source wrapper that provides full access to the basic and enhanced capabilities of both servers. This public API is a component of caCORE.

LexBIG API

LexBIG is an open-source, public domain vocabulary server being developed by the Mayo Clinic. It is the source of content for the NCI BioPortal browser (described on page 10). LexBIG uses a well-structured application programming interface (API) capable of accessing and distributing vocabularies as commodity resources.

The LexBIG API provides the following key features:

- A robust, scalable, open-source implementation of EVS-compliant vocabulary services. The API specification was based on, but not limited to, fulfillment of the caCORE EVS API. This specification will be further refined to accommodate changes and requirements based on prioritized needs of the caBIG™ community.

- A flexible implementation for vocabulary storage and persistence, allowing for alternative mechanisms without affecting client applications or end users. Initial development will focus on delivery of open-source, freely available solutions, though it will not preclude the ability to introduce commercial solutions such as Oracle.
- Standard tools for load and distribution of vocabulary content. This includes, but is not limited to, support of standardized representations such as the UMLS Rich Release Format (RRF), OWL (Web Ontology Language), and Open Biomedical Ontologies (OBO).

Other EVS Terminology Resources

NCI Terminology Browser

The open-source NCI Terminology Web browser provides direct access to a number of biomedical terminologies of special interest. In addition to the NCI Thesaurus, its terminologies include SNOMED CT, MedDRA, LOINC, VA NDF-RT, GO, and the MGED Ontology.

FDA Terminology Data Standards

EVS is working with the U.S. Food and Drug Administration (FDA) to develop and support controlled terminology in several areas, including Structured Product Labeling (SPL). SPL is a document markup standard approved by the Health Level Seven (HL7) organization and adopted by the FDA as a mechanism for exchanging information about medications.

NCI BioPortal

The *NCI BioPortal* is a terminology browser that will ultimately replace the NCI Terminology and Metathesaurus browsers. You can use the BioPortal to browse, search, and submit a new term request.

The NCI BioPortal serves several key functions:

- **Browse Tool for LexBIG Data:** The BioPortal enables you to browse the contents of the LexBIG library. When browsing, you can view project information and available versions. You can also generate tree and graph views for selected ontologies.
- **Search Tool for LexBIG Data:** The BioPortal provides two methods for searching terminology content: *quick search* and *advanced search*.
    - With *quick search*, you can search the LexBIG library across all ontologies, within certain terminologies, or in a specific terminology. You can search by class name, ID, or attributes such as synonyms.
    - With *advanced search*, you can search the content of a single terminology using an additional role or association.

- **Term Submission Tool:** The BioPortal provides a link from which you can submit a new term request for consideration by EVS staff. The New Term Request link leads to a GForge Tracker page where you can view and browse a list of previously submitted terms to ensure that your suggested term has not been previously submitted. If your term is not in the list, you can link to a submission form, enter details about the proposed new term, and submit the request.

CHAPTER

2

DESCRIPTION LOGIC AND THE NCI THESAURUS SEMANTIC MODEL

This chapter provides an overview of knowledge representation, description logic, and the use of description logic in the NCI Thesaurus.

Topics in this Chapter

- *About Knowledge Representation* on this page
- *About Description Logics and OWL* on page 15
- *Description Logic in the NCI Thesaurus* on page 23

About Knowledge Representation

Frame-Based Representations

Knowledge representation has long been a prime focus in artificial intelligence (AI) research. This area of research asks how we can accurately encode the rich and highly detailed world of information that is required for the application area being modeled and yet, at the same time, capture the implicit commonsense knowledge. One of the most common approaches to this problem in the 1970s was to use *frame-based representations*.

The basic idea of a frame is that important objects in our world fall into natural classes, and that all members of these classes share certain properties or attributes, called *slots*. For example, all dogs have four legs, a tail (or vestige of one), and whiskers. Restaurants generally have tables, chairs, eating utensils, and menus. Thus, when we enter a new restaurant or encounter a new dog, we already have a *frame of reference* and some expectations about the properties and behaviors of these entities.

In a seminal paper by Marvin Minsky published in 1975, the author placed the frame representation paradigm in the context of a semantic network of nodes, attributes, and relations. *Figure 2.1* shows a simple frame-based representation of an earthquake as it might be used in a semantic network of news stories.[1]

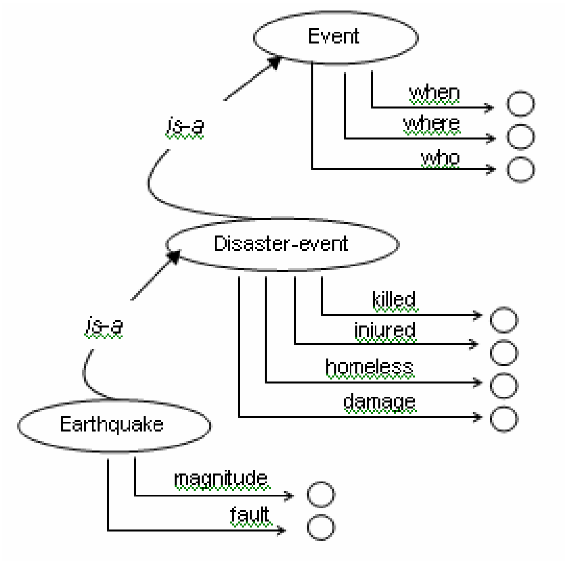


Figure 2.1 An Earthquake in a semantic network of news stories

First-Order Predicate Logic

At the same time that frame-based representations were being explored, a popular alternative approach was to use some subset of *first-order predicate logic* (FOL), often implemented as a Prolog program. While propositional logic allows us to make simple statements about concrete entities, a complete first-order logic enables us to make general statements about anonymous elements, with the introduction of variables as placeholders. *Table 2.1* contrasts the difference in expressivity between propositional logic and FOL.

Propositional Logic	***First-Order Predicate Logic***
All men are mortal.	$\forall x$: Man (x) => Mortal(x)
Socrates is a man.	Man(Socrates) => Mortal(Socrates)
Socrates is mortal.	

Table 2.1 Propositional logic vs. first-order predicate logic

In other words, using FOL, you can express general rules of inference that can be applied to all entities whose attributes satisfy the left-hand side of the inference → operator. Thus, simply asserting *Man*(Socrates) entails *Mortal*(Socrates).

Since logic programming is based on the tenets of classical logic and comes equipped with automated theorem-proving mechanisms, this approach enabled the development of inference systems whose soundness and completeness could be rigorously demonstrated. But while many of these early inference systems were logically sound and complete, they were often not very useful, as they could only be applied to highly proscribed areas, or "toy problems." The problem was that a complete first-order

---

[1] Patrick Winston, *Artificial Intelligence* (Massachusetts: Addison-Wesley, 1984).

predicate logic is itself computationally intractable, as certain statements may prove *undecidable*.

Suppose, for example, that we are trying to establish that some theorem, *P(x)*, is true. The way a theorem prover works is to first negate the theorem and, subsequently, to combine the negated theorem ($\neg P(x)$) with stored axioms in the body of knowledge to show that this leads to a logical contradiction. Ultimately, when the theorem prover derives the conclusion that $P(x) \wedge \neg P(x)$ is inconsistent—that it results in the null set—the program terminates and the theorem is considered proven.

This method of proof by refutation is guaranteed to terminate when it is indeed upheld by the body of knowledge. The problems arise when the initial theorem is not valid, as its negation may not produce a logical contradiction, and thus the program may not terminate.

In contrast, frame representations offered a rich, intuitive means of expressing domain knowledge, yet they lacked the inference mechanisms and rigor that predicate logic systems could provide. As suggested by *Figure 2.1* on page 14, the frame representation captures a good deal of implicit knowledge. For example, we expect that all disaster events, including earthquakes, have information about fatalities and injuries and the extent of loss and property damage. In addition, we expect that these events will have locations, dates, and individuals associated with them.

Early efforts to apply predicate logic to frame representations to make this information explicit, however, soon revealed that the problem was computationally intractable. This occurred for two reasons: (1) The frame representation was too permissive; more rigorous definitions were required to make the representation computational; and (2) first-order predicate logic itself was computationally intractable.

Several subsets of complete FOL have since been defined and successfully applied to develop useful computational models capable of significant reasoning. For example, the Prolog programming language is based on a subset of FOL that severely limits the use of negation.

The family of *description logic* (DL) systems is a more recent development. Because these systems function as an auto-classifier, they are especially well-suited to the development of ontologies, taxonomies, and controlled vocabularies.

About Description Logics and OWL

Description Logics (DLs) are a family of languages that can be used to represent terminological systems. A DL is an extension of the frame-based knowledge representation formalism but with defined semantics based on set theory. It is a decidable subset of first order logic and can be viewed as a combination of the frame-based approach with FOL. Like frames, the DL representation allows for concepts and relationships among concepts, including simple taxonomic relations and other meaningful types of association.

Today, there is a large family of description logics that have been realized with varying levels of expressivity and resulting computation complexities (*Table 2.2* on page 16). The difference among DLs boils down to the construction operations allowed by the specific type of DL. For example, the most minimal form of a DL does not allow any form of existential quantification. This limitation allows for a very easily computed solution space, but the resulting expressivity is severely diminished. Reasoners (classifiers) can be designed to deal only with the specific constructors of that DL, therefore simplifying computational effort. The theory behind these models is beyond the scope of this discussion. If you are interested in further exploring this theory, read *The Description Logic Handbook* by Franz Baader, et al. (Eds.), Cambridge University Press, 2003, ISBN 0-521-78176-0.

Table 2.2 describes various types of DL expressivities.[2]

Abbreviation	***Description***
$\mathcal{AL}$	Attributive language. This is the base language that allows atomic negation, concept intersection, universal restrictions, and limited existential quantification.
$\mathcal{FL}$-	A sub-language of $\mathcal{AL}$ that disallows atomic negation.
$\mathcal{FL}_o$	A sub-language of $\mathcal{FL}$- that disallows limited existential quantification.
$\mathcal{C}$	Complex concept negation.
$\mathcal{S}$	An abbreviation for $\mathcal{AL}$ and $\mathcal{C}$ with transitive properties.
$\mathcal{H}$	Role hierarchy (sub properties - rdfs:subPropertyOf).
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.
$\mathcal{O}$	Nominals (enumerated classes of object value restrictions - owl:oneOf, owl:hasValue).
$\mathcal{I}$	Inverse properties.
$\mathcal{N}$	Cardinality restrictions (owl:Cardinality, owl:MaxCardinality).
$\mathcal{Q}$	Qualified cardinality restrictions (available in OWL 1.1).
$\mathcal{F}$	Functional properties.
$\mathcal{E}$	Full existential qualification (existential restrictions that have fillers other than owl:thing).
$\mathcal{U}$	Concept union.
$(\mathcal{D})$	Use of data type properties, data values, or data types.

Table 2.2 DL expressivities

The Web Ontology Language (OWL) has been designed to facilitate the *semantic* aspect of the Semantic Web. OWL is built on top of RDF/RDFS, an XML-based data representation scheme.

---

[2] *DL Expressivity*, from Wikipedia
(April 2007: http://en.wikipedia.org/wiki/Description_logic#DL_Expressivity)

OWL has three levels of expressivity:

- *OWL Lite* is computationally tractable but is not expressive enough to represent the information content of the NCI Thesaurus, and thus we will not consider it further.
- *OWL Full* provides the full representation power of RDF/RDFS but is not guaranteed to be decidable. That is, a terminology constructed in OWL Full may not be classifiable.
- *OWL DL* has restrictions on the expressions that can be asserted and is decidable; it is a $\mathcal{SHOIN}(\mathcal{D})$ DL (*Table 2.2* on page 16). Although the NCI Thesaurus is built as an OWL DL (originally as an $\mathcal{FL}$- in Ontylog), some of the constructions we use on conversion to OWL cause it to validate as an OWL Full ontology. However, these OWL Full constructions are limited to annotation properties (such as the Synonym that is declared with a range restriction of *string*) and are ignored by the classifier. Semantic constructions that would require the NCI Thesaurus to become OWL Full would not be acceptable. *Table 2.3* on page 23 describes the expression syntax used in OWL.

For more information about the three levels of expressivity, see http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3.

Concepts and Roles

The basic entities in a DL are the *concept* and the *role*:

- *Concepts* in DL are defined in terms of sets of *individuals*. Because DL has set-theoretical semantics, keep individuals in mind when modeling concepts, even in the case of a structure such as the NCI Thesaurus where individuals are not part of the terminology.
- *Roles* are binary relations between concepts. Note that a qualified role assertion on a class defines an anonymous concept (the unnamed set of individuals making up the class—the DL view of *concept*).

A subtle distinction exists between a DL concept and a terminologic concept: the intensional meaning of a class is the underlying terminologic concept, but the class extension is the collection of individuals making up the class. Hence, two classes could have the same class extension yet represent different concepts.

OWL uses a slightly different terminology for DL entities: a concept is denoted a *class*, and a role a *property restriction*. The Protégé/OWL GUI interface follows this convention. To maintain a *lingua franca*, we are moving towards adopting all the elements of the terminology in NCI Protégé; however, some terms might show vestiges of the previous editing environment during a transition period.

Note: Be aware of the differences in terminology being discussed here. If you research the DL or OWL literature, you will encounter one or the other. In our context, concept and class, as well as role and property restriction, can be used interchangeably.

OWL Class Descriptions and Anonymous Classes

A DL representation is constructed from a base set of *primitive concepts*, which are simply concepts asserted with a *necessary* description. *Defined concepts* with *necessary and sufficient* conditions are then derived from these primitive concepts using the constructors allowed by that DL, such as role assertions, intersections, and unions.

OWL provides six different concept construction/description mechanisms:

1. Identification (for example, named classes such as `ras_gene`)
2. Property restrictions
3. Enumeration
4. Intersection
5. Union
6. Complement

The first mechanism allows you to create a class by declaring it with a name or identifier. The other five result in the creation of unnamed *anonymous* classes. These are used in assertions of *necessary* or *necessary and sufficient* conditions that describe or define a class and constrain the individuals that make up the class. For example, when applied to a named class such as *Oncogene_TIM*, the property restriction

> *some Gene_Plays_Role_In_Process Signal_Transduction*

essentially means that individuals that make up the *Oncogene_TIM* class describe a subset of the set of individuals that have the relation *Gene_Plays_Role_In_Process Signal_Transduction*. This set is an anonymous class.

For another example, consider a class *AnatomicParts* consisting of individuals that are parts of other entities in an Anatomy domain. This class is illustrated by the left drawing in *Figure 2.2* on page 19. If the "part" relationship is expressly asserted as a "must have" property on the individuals in this class, the *AnatomicParts* class could also have been left unnamed and referenced anonymously as the *some part_of Anatomy* class, as shown by the right drawing in the figure.

Very often, however, a class such as *AnatomicParts* will be created as the named class and defined with *some part_of Anatomy* as one of its necessary and sufficient conditions (discussed below). The individual instances of this class must also be described by the *part_of* property restriction, with either the Anatomy class or one of its subclasses as the filler value.

For reference, four-pointed stars in *Figure 2.2* represent individuals, and circles represent classes (sets of individuals). Red arrows indicate relationships between individuals, and the backward letter E (∃) represents the existential qualifer.

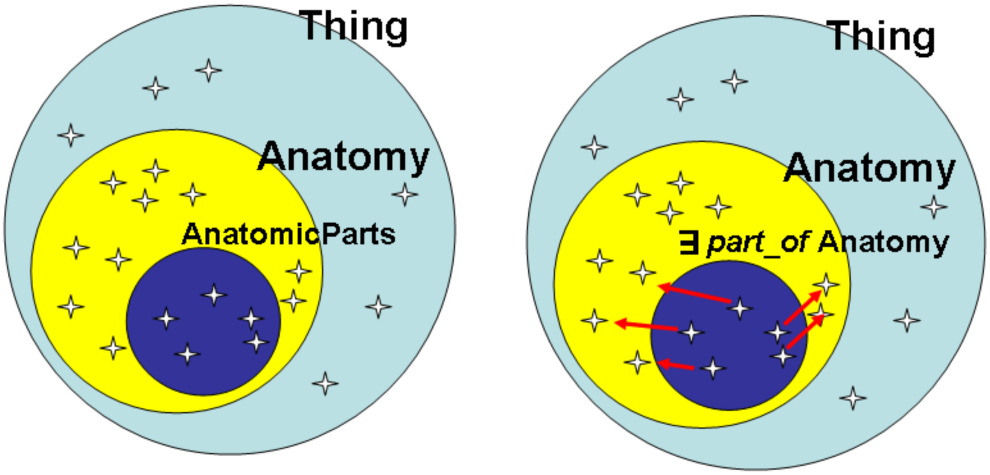


Figure 2.2 "Thing" OWL class and some subtypes of the anatomic domain

Other anonymous classes can be described with the previously mentioned constructors. The *intersection* is used implicitly when a class has a number of conditions asserted, such as two property restrictions. But it can also be asserted explicitly when one condition is a class expression. In terms of individuals, the anonymous class representing the intersection of classes is defined by the individuals that are members of all the listed classes in the construction. In NCI Protégé, intersections are not directly supported; intersections are used, however, in class expressions termed *role groups* (discussed below), but the Extension does not make the details of the construction explicit to the editors.

As with the intersection, NCI Protégé does not support the union constructor directly; that is, you cannot construct at will a class expression consisting of a union of classes. The union is used, however, in the construction of role groups. In terms of individuals, the anonymous class described by a union of classes is defined by the individuals that are members of any of the classes listed in the union.

The *complement* constructor (NOT operator) is currently not supported by NCI Protégé, but it will be considered for future use. It is analogous to logical negation, and it is used to describe a class defined by the set of individuals NOT in the class that is the object of the complement assertion, such as

sick = complementOf(healthy)

Because a number of concepts in the NCI Thesaurus deal with abnormal things, stating this formally in the terminology might be beneficial. However, negation must be asserted carefully, as the result of a description containing the complement constructor might not reflect your intentions. This is illustrated in *Figure 2.3* on page 20.

Suppose, for example, that the narrative of an experimental observation or a clinical finding includes the description that "the liver is not affected." Further, the terminology of the anatomy domain is constructed such that the parts of liver are identified, and the editor intends to include parts of liver in the formal description or definition of the observation-concept or finding-concept so that end users can code artifacts with it. The concept will likely have additional conditions asserted; however, for the present argument we need only consider the "parts of liver" portion.

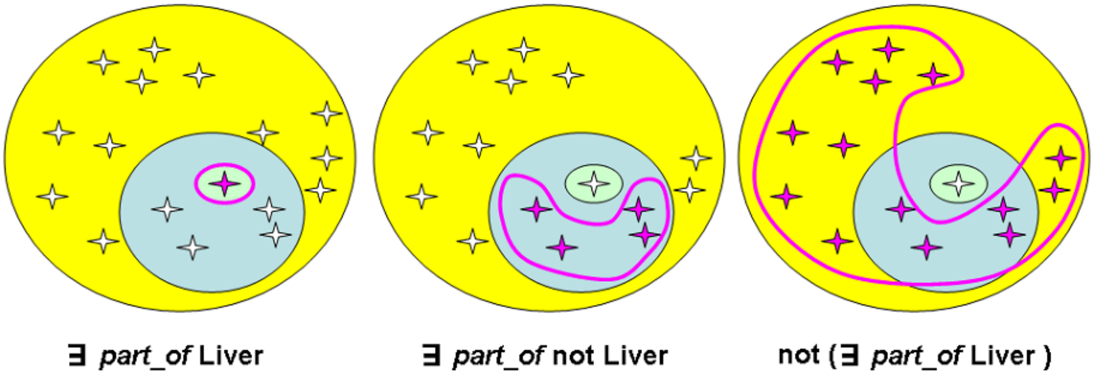


Figure 2.3 Effect of complement constructor on anonymous classes described by part_of role with domain anatomicpart and range anatomy

For reference, in *Figure 2.3* the outermost yellow circle represents the Anatomy class, the light blue middle circle represents the class *AnatomicParts*, and the light green innermost circle represents the class of individuals that are a *part_of* Liver. Anonymous classes of interest are bounded by a pink line: the left drawing depicts the anonymous class described by the property restriction *some part_of Liver*. In the center drawing, the condition *some part_of (not Liver)* describes the set of individuals in the *AnatomicParts* class that are not part of Liver, while the drawing on the right shows the set of individuals in the Anatomy class that are not parts of Liver (but would include Liver as a whole) as described by the condition *not(some part_of Liver)*.

If the Anatomy terminology is a portion of a broader terminology, concepts outside the anatomy domain, such as *Bicycle* and *Mouse*, would also be included in this last anonymous class. Depending on where in the expression the complement is asserted, the class could include individuals or classes that the editor did not intend to include; not only would additional constraints on class membership be required, but care would be needed in the specification of these constructions.

An expression such as *not(Liver)* is termed *atomic negation*. The complement of more involved class expressions such as *not(some part_of Liver)* is referred to as *complex negation*. Examining the various cases in the NCI Thesaurus where complex negation is beneficial would perhaps allow the development of interface support for class expressions containing the complement constructor. This would minimize semantic errors in their specification. If this is not feasible, negation might be restricted in the NCI Thesaurus.

The final class description mechanism using the *enumeration* constructor describes and defines a class by the exhaustive enumeration of its individual instances. Because the NCI Thesaurus does not include individuals, NCI Protégé does not support this mechanism.

Subsumption and Disjointness Axioms

Classes can be described by *necessary* assertions on the class. These primitive classes are often used at the top levels of a hierarchy to provide a structure suited for a particular purpose. The *subsumption* axiom is asserted on these classes as part of their description (as *subclasses* in OWL), essentially describing the asserted hierarchy. For individuals, the subsumption axiom means that if class X is a subclass of Y, then the individuals making up class X are a subset of the individuals in class Y; in other words, you cannot construct a class with individual instances that are not also individuals of the parent class, class Y subsumes X.

A second axiom that can be asserted on OWL classes is the *disjointness* axiom. Again in terms of individuals, if class X and Y are disjoint, then any individual in class X cannot also be a member of class Y, and vice-versa. The disjointness axiom can be asserted on a class as part of its description. As with the subsumption axiom, it is a partial definition; it imposes a necessary but not sufficient condition on class membership.

In the case of the NCI Thesaurus, disjointness has only been stated at the uppermost level of the domain hierarchies (the so called *Kinds*), but this can be expected to change in the future as it provides useful constraints on classes, such as when asserting property restrictions with the universal quantifier.

Equivalency and Necessary vs. Necessary and Sufficient Conditions

OWL provides a third axiom applicable to classes: the *equivalency* axiom (*equivalentClass* in OWL speak). This axiom allows classes to be declared *defined*; that is, the collection of asserted conditions is *necessary and sufficient* to indicate whether an individual is a member of a class. This means that equivalent classes have the same exact set of individuals.

It is worth mentioning that the notion of class *equality* is different from that of class *equivalency*; two classes are *equal* if they denote the same conceptual entity, not only that they are made up by the same collection of individuals. (Equality cannot be stated in OWL DL—only in OWL Full). In the Protégé OWL environment, stating that a class has necessary and sufficient conditions indicates that the class is defined, which means that it is equivalent to the collection of assertions on the class.

Note: The equivalent class axiom does not need to be asserted in the case of named classes described by an enumeration of its members (the third construction/description mechanism listed above). The enumeration defines class membership.

In a *primitive class* (also termed a *partial* class, necessary conditions asserted) any individual that is a member of that class must satisfy the asserted conditions. In a *defined* class (also termed a *complete* class, necessary and sufficient conditions asserted), not only must the individuals of that class satisfy the conditions, but any other individual that meets those conditions must then also be a member of the class. When applied to classes (sets of individuals), subsumption checking results in the retreeing of classes to generate an inferred hierarchy. This is represented in *Figure 2.4*, which depicts classes A, B, C, D, E in a hierarchy: B and C are subclasses of A, E is a subclass of C, D is a subclass of B, and C is a defined class.

In simple terms, when a class such as A is *primitive*, class A can ask of its subclasses B, and C "do you meet my conditions for membership?" In contrast, when a class is *defined*, such as C, C can ask this same question from E, but E can also ask from C *"*do I meet your conditions for membership?" However, this doesn't apply only to E. For instance, D could ask the same question from C, and if D met C's conditions, it would be inferred that D is a subclass of C as well.

In *Figure 2.4*, solid lines represent the subclass relation in an upward direction. For example, D is a subclass of B. The dotted line represents a potential subclass relation. The arrows represent a class membership query.

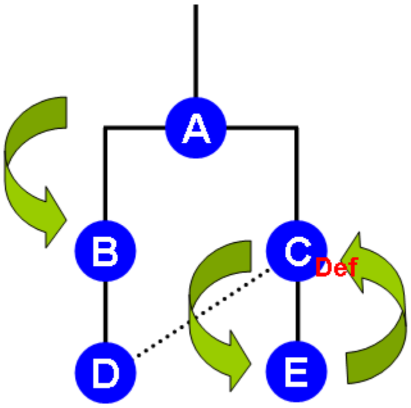


Figure 2.4 Hierarchy of named classes A, B, C, D, E, with Class C as defined

Property Restrictions and the Existential and Universal Qualifiers

OWL permits two types of property restrictions: *cardinality constraints* and *value constraints.* Cardinality constraints (cardinality equal to a number, or a minimum cardinality, or a maximum cardinality) either specify or limit the number of times that a property can be asserted on a class. The value constraints *some*, *all*, and *has value* place restrictions on the range of a property when applied to a class.

The *some* value constraint, also known as the *existential quantifier* and represented by a backwards E (∃), can be read as at least one. For example, the property and filler value

some Disease_Has_Finding Amorphous_Eosinophilic_and_Acellular_Deposit

says that an individual of this class must have **at least one** property, *Disease_Has_Finding* with the value *Amorphous_Eosinophilic_and_Acellular_Deposit*. This individual can have *Disease_Has_Finding* multiple times with different filler values (or as restricted by a cardinality constraint), but at least one of these times it must have the *Amorphous_Eosinophilic_and_Acellular_Deposit* value (or a subtype). That is, an individual such as *disease_from_patient_X* can have multiple disease findings, but it must also have the specified finding to be in the class.

The all constraint, also known as the *universal quantifier*, is represented by an inverted A (∀). This constraint can be read as **only**. For example,

all Disease_Has_Finding Amorphous_Eosinophilic_and_Acellular_Deposit

says that **if** the property *Disease_Has_Finding* exists in an individual of this class, it can only have the value *Amorphous_Eosinophilic_and_Acellular_Deposit*. (The property need not be specified for an individual member of the class unless this condition is present.) In terms of the individual *disease_from_patient_Y*, it can only have the specified finding or no finding at all. However, because OWL does not make a unique name assumption on classes, asserting this property multiple times with different values on the same class results in an error only if the filler value classes are explicitly made disjoint elsewhere in the terminology.

Currently the NCI Thesaurus does not make use of the *has-value* restriction on properties (which must have either an individual or data value as the filler). It would, however, allow simplification of some domains containing concepts that need not exist in a terminology, such as concepts representing nucleotide positions of genes. NCI Protégé supports *has-value* restrictions, but at the present time a software component named DIG, an interface between Protégé and reasoners such as FaCT++ and Racer, does not support this type of restriction.

Description Logic in the NCI Thesaurus

The NCI Thesaurus was previously developed using the proprietary Ontylog™ implementation of description logic from Apelon, Inc. It was edited and maintained in the Terminology Development Environment (TDE) provided by Apelon. The TDE is an XML-based system that implements the DL model of description logic based on Apelon’s Ontylog Data Model.

The Thesaurus is now edited and maintained using Protégé, an open-source development environment for ontologies and knowledge-based systems. Protégé includes a plug-in that supports the Web Ontology Language (OWL). Both Protégé and the plug-in were developed at Stanford Medical Informatics (SMI).

The NCI Center for Bioinformatics (NCICB) has developed NCI Protégé, a tab plug-in that extends Protégé to support the NCI-specific editing environment. To facilitate use of NCI Protégé, the Ontylog Description Logic (DL) used by Apelon’s TDE was converted to OWL.

Table 2.3 provides a guide to OWL expression syntax used in Protégé.[3]

OWL Element	***Symbol***	***Key***	***Example***
allValuesFrom	∀	*	∀ children Male
someValuesFrom	∃	?	∃ children Lawyer
hasValue	□	$	rich □ true
cardinality	=	=	children = 3
minCardinality	≥	>	children ≥ 3
maxCardinality	≤	<	children ≤ 3
complementOf	¬	!	¬ Parent
intersectionOf	∩	&	Human ∩ Male

Table 2.3 Owl expression syntax used in Protégé

---

[3] Holger Knublauch, *Editing Semantic Web Contents with Protégé: The OWL Plugin* (International Workshop on Description Logics, Whistler, BC, Canada, 2004: http://www.knublauch.com/publications.html).

OWL Element	Symbol	Key	Example
unionOf	∪	|	Doctor ∪ Lawyer
enumeration	{...}	{ }	{male female}

Table 2.3 Owl expression syntax used in Protégé (Continued)

Tip: OWL is an extensive language and is outside of the scope of this guide. Although several useful OWL tutorials are available online, the famous "pizza tutorial" is the most significant and comprehensive. You can download the pizza tutorial at http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf.

OWL and the NCI Thesaurus

In 2005, Stanford Medical Informatics (SMI) conducted an analysis of the NCI Thesaurus, considering modeling patterns as well as use case requirements. After consulting with SMI, members of the EVS agreed to implement certain recommendations as part of the transition, while deferring other recommendations for the future. NCI Protégé was designed accordingly.

The rest of this section discusses specific recommendations of the SMI analysis.

Identifiers

OWL is built using the Resource Description Framework (RDF), which the World Wide Web Consortium (W3C) describes as "...particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page..." *RDF schema*, or *RDFS*, is RDF's vocabulary description language.

The EVS team agreed to eventually use the concept *code*, rather than the concept *name*, as an identifier, and to make the code non-editable. Protégé uses the OWL *rdf:ID* property to represent the concept code, and it uses the *rdfs:label* property to display the value of the *Preferred_Name* property.

Class Expressions

Although creating role groups in Protégé is somewhat awkward, the Protégé interface needs to support them. The EVS team agreed that role groups represented in Ontylog are best represented in OWL with *OR* semantics. Once NCI editors are fully operational with Protégé, they will most likely have to manually validate role groups.

Property Hierarchy

Subproperties do not always override inherited parent properties. While you can use them in the OWL version of the NCI Thesaurus, you cannot use them in the same way that you would use role hierarchies in Ontylog to override inherited values outside the domain of the inherited, asserted filler value.

Property Inverses

Property inverses are not always applicable. They require modeling for such things as relations between genes and gene products. Even when applicable, you must manually assert them to diminish the possibility of inappropriate usage.

Datatype Properties

We might eventually use datatype properties for chromosomal location or bands, which require the use of *hasValue*, rather than *someValuesFrom*. These properties will not be used immediately, however, because they would require that existing chromosomal locations/band concepts be retired.

Transitivity of Properties

Transitivity applies to *Part_Of* and *Has_Location* roles. You can manually apply transitivity during migration cleanup.

More Precise Declaration of Property Domains and Ranges

When modeling a domain, editing groups should conduct an analysis to ensure that domains and ranges are precise. For existing roles, editors can establish a more precise declaration of the range after the transition to using OWL.

Some vs. All Property Restrictions

The *All* role modifier has not been properly applied in Ontylog, and its use needs to be corrected. The semantics of *All* are the same in Ontylog and OWL, but the EVS team decided to conduct a more detailed review of its usage after making the transition to using OWL.

For modeling in other domains, the use of *All* is not preferred, because it is not straightforward, easy to use, nor easy to understand.

Defining and Non-Defining Restrictions

To separate restrictions into *Necessary* vs. *Necessary & Sufficient*, editing groups will need to identify *defining* roles for specific domains.

Disease-Anatomy Relationships

The SMI analysis recommended that relations between the disease and anatomy domains be re-examined. They recommended that, in some cases, we consider expressing them differently.

For example, we might define the anatomic site for *Gastrointestinal Carcinoma* as *Gastrointestingal System* or any of its parts:

> *Gastrointestinal Carcinoma Disease_Has_Associated_Anatomic_Site some (Gastrointestinal_System or (some Anatomic_Strcture_is_Physical_Part_of Gastrointestinal) System))*

Although this is not critical, we need to better understand how this change might affect retrieval applications.

Property Qualifiers

Though property qualifiers don't exist in OWL, they can be implemented in various ways. Currently, we can implement them using reification, or we can implement them as *complex properties* using pipes or pseudo-XML to delimit fields.

After discussing this with SMI and testing it in the Protégé OWL environment, the EVS team decided against reification as a viable alternative for FULL_SYNs in the Thesaurus. We do realize that we need to eliminate complex property values and to continue to examine native OWL options.

Associations

These are non-inheritable relations between concepts. You can view them as an annotation of a class with another class, which you can express as *ObjectAnnotation* properties.

Exceptional Conditions

You can override a "normal" value with an exception by converting an *exclude* role in Ontylog to a negation in OWL. You can perform negation on the expression or the value, with different consequences.

Overriding inherited conditions through negation is problematic, however. We will need to re-examine the use of exclude roles in the disease domain.

CHAPTER
3
GETTING STARTED WITH PROTÉGÉ

This chapter explains where to find the latest NCI Protégé application files, install a new build, and log in. It also explains how to install a local NCI Thesaurus subset file in case you want to work with NCI Protégé offline.

Topics in this Chapter

- *Working with NCI Protégé Application Files* on this page
- *Logging In to the Protégé Server* on page 30
- *Using a Local Thesaurus Subset* on page 32

Working with NCI Protégé Application Files

NCI Protégé installation files and other related files are available for download on GForge. They are listed in the Files area of the *NCICB EVS Collaborative Terminology Development Tools* project at http://gforge.nci.nih.gov/projects/protegegui/.

Some of the available files are for developers only. *Table 3.1* lists the files that are relevant to NCI editors.

Category	***Subcategory/File name***	***Purpose***
Base Installation	Production Client Build: `Protege.Client.BGT-1.3.x.exe` (Windows users) `Protege.Client.BGT-1.3.x.zip` (Mac users)	Installs the NCI Protégé application. The `.exe` version unzips automatically when you run it on a Windows machine, but it does not run on a Mac. If you are a Mac user, download and install the `.zip` version. **Note:** The BGT version is for BiomedGT, and the NCIt version is for the NCI Thesaurus.

Table 3.1 Installation files for NCI Protégé

Category	***Subcategory/File name***	***Purpose***
Documentation	Release Notes Beta 1: `Beta_1_ReleaseNotes.doc`	Provides an overview of new features, known bugs, bug fixes, limitations, and workarounds in the newest release. The release notes also include any documented information written after manuals were released.
OWL File (Optional)	NCIT Subset 060926 w Retired: `Thesaurus-ByName-060926-Retired.zip`	Provides a subset of the NCI Thesaurus that you can install on your hard drive and use as a practice file. For more information, see *Using a Local Thesaurus Subset* on page 32.

Table 3.1 Installation files for NCI Protégé (Continued)

Downloading the Latest Installation Files

The NCI Protégé application files are packaged in one installation file (`.exe` or `.zip`). This is the only file you need to install the current build.

To download the base installation files for NCI Protégé, follow these steps:

1. Log in to GForge at http://gforge.nci.nih.gov/.
2. Locate the **NCICB EVS Collaborative Terminology Development Tools** project at http://gforge.nci.nih.gov/projects/protegegui/.
3. Click the **Files** tab.
4. Select one of the following files under the **Base Installation** category:

   Windows users:

   - `Protege.Client.BGT-1.2.x.exe` or
   - `Protege.Client.NCIt-1.2.x.exe`

   Mac users:

   - `Protege.Client.BGT-1.2.x.zip` or
   - `Protege.Client.NCIt-1.2.x.zip`
5. Save the file to a temporary directory on your hard drive.
6. Note the name of the directory in which you stored the files.

Note: The procedures in this guide refer to the Windows environment. If you are a Mac user, consult the operating system help for information about installing applications from zipped files.

Uninstalling Previous Versions of NCI Protégé

Different versions of NCI Protégé can co-exist on the same machine, but you can easily remove an unused version to save disk space.

To uninstall a previous version of Protégé, follow these steps:

1. Using Windows Explorer, locate either of the following directories, depending on which version of the fil you installed:
    - `C:\Program Files\Protege.Client.BGT-1.2.x` or
    - `C:\Program Files\Protege.Client.NCIt-1.2.x`

    Note: The name of the application folder includes the current build number.
2. Delete the entire directory.

    You do not have to delete related Windows Registry entries.

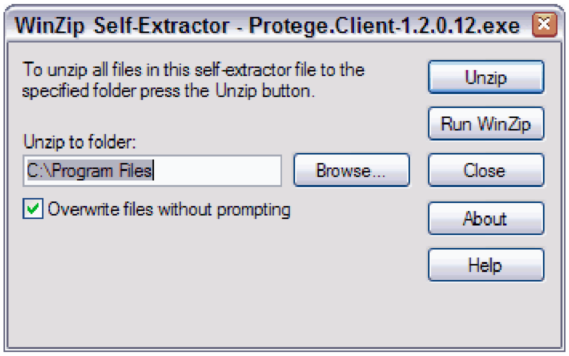
Installing a New Build

Prerequisite procedure: *Downloading the Latest Installation Files* on page 28

To install the NCI Protégé application, follow these steps:

1. Locate the `ProtegeClient-1.2.x.exe` file on your hard drive, then double-click the file name.

    The WinZip Self-Extractor window opens (*Figure 3.1*). The Self-Extractor automatically unzips the files to the `C:\Program Files` directory.



WinZip Self-Extractor - Protege.Client-1.2.0.12.exe

To unzip all files in this self-extractor file to the specified folder press the Unzip button.

Unzip to folder: C:\Program Files

☑ Overwrite files without prompting

Figure 3.1 Winzip Self-Extractor window - new installation

2. Ensure that the **Overwrite files without prompting** box is checked, then click **Unzip**.
3. When the WinZip Self-Extractor message window opens, click **OK** to close the window.
4. Click **Close** to close the WinZip Self-Extractor window.
5. Using Windows Explorer, confirm that the following directory was created on your hard drive: `C:\Program Files\Protege.Client-1.2.x`

    Note: The name of the application folder includes the current build number.

Logging In to the Protégé Server

To log in to NCI Protégé, you need a user name, a password, and the host machine name. If you do not already have this information, contact the NCI Protégé administrator.

To start NCI Protégé and connect to the database, follow these steps:

1. In Windows Explorer, navigate to the following location, depending on which version of the application is installed:
   - `C:\Program Files\Protege.Client.BGT-1.2.x` or
   - `C:\Program Files\Protege.Client.NCIt-1.2.x`

   Note: The name of the application folder includes the current build number for NCI Protégé.
2. Double-click the following file: `run_protege.bat`.

   A Console (command line) window opens, followed by two other windows:
   - The Protégé main application window, and
   - The Welcome to Protégé window, displayed in front.
3. In the Welcome to Protégé window, select **Open Other** (*Figure 3.2*).

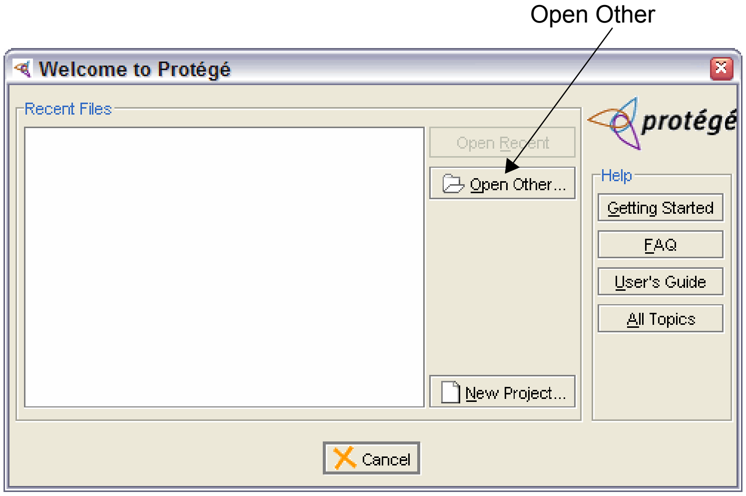
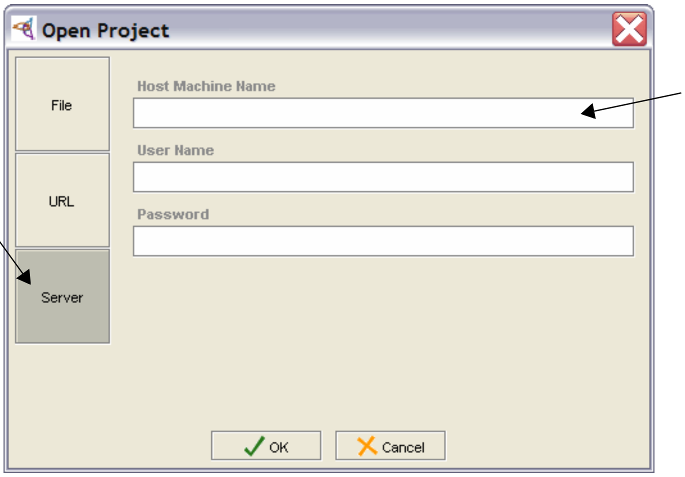


Figure 3.2 Welcome to Protégé window

4. In the Open Project window, click **Server** if it is not already selected (*Figure 3.3* on page 31).
5. Enter the **Host Machine Name** if necessary.

   Note: If you do not know the host machine name, contact your NCI Protégé administrator.



Click Server...

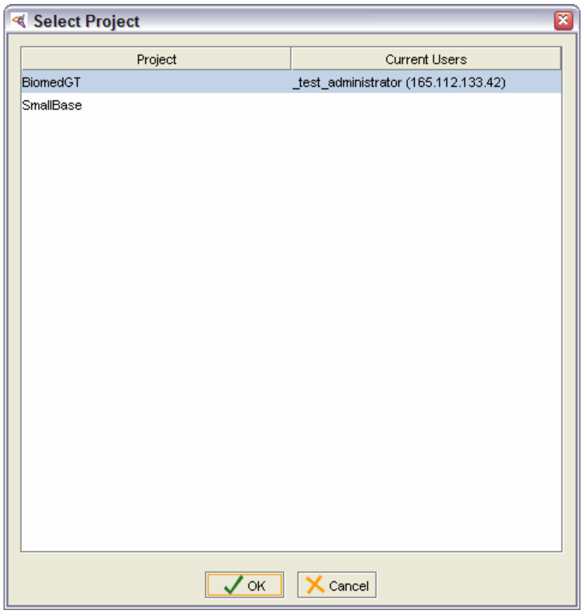
...then ensure that the Host Machine Name shows here.

Figure 3.3 Open Project window - server button

6. Enter your user name and password, then click **OK**.

   The Select Project window opens (*Figure 3.4*).

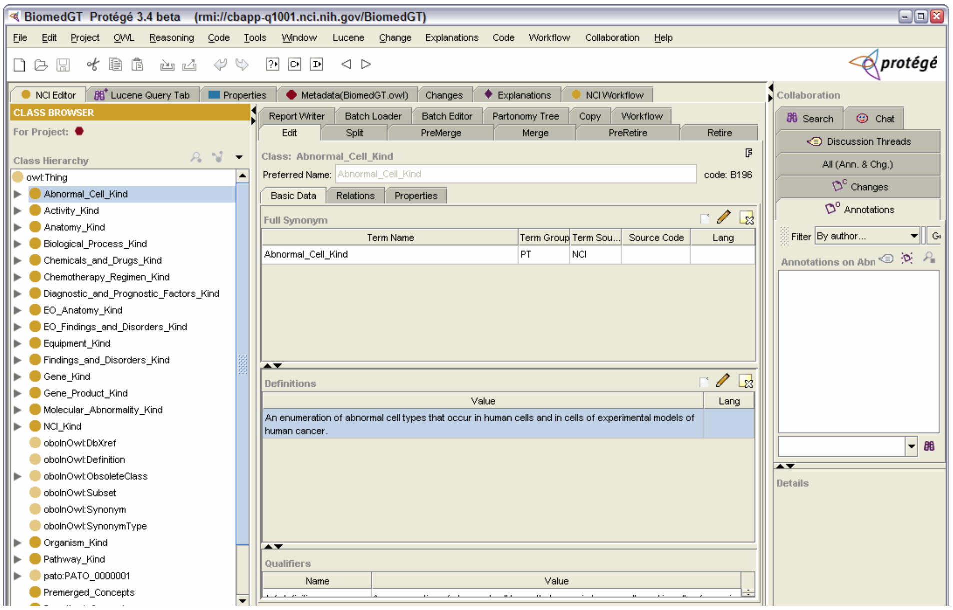
7. Select the **BiomedGT** project, then click **OK**.



Project	Current Users
BiomedGT	_test_administrator (165.112.133.42)
SmallBase	

Figure 3.4 Select Project window

The main NCI Protégé window opens (*Figure 3.5*).



Class: Abnormal_Cell_Kind

Preferred Name: Abnormal_Cell_Kind code: B196

Term Name	Term Group	Term Sou...	Source Code	Lang
Abnormal_Cell_Kind	PT	NCI		

Value	Lang
An enumeration of abnormal cell types that occur in human cells and in cells of experimental models of human cancer.	

Figure 3.5 Main NCI Protégé window

Using a Local Thesaurus Subset

As an NCI editor, you normally work with NCI Protégé while connected to a server. If you want to use the application without using the production database, you can install a local subset of the NCI Thesaurus on your hard drive. You can then use this file to create a local project for working with test data and learning the Protégé interface.

When you first open a local file in NCI Protégé, the NCI-specific tabs are not displayed. You need to change the local tab configuration to match the tab display used on the Protégé server.

Once you set up the tabs and the tab order, you can save the configuration as a *project*. The project file is stored on your hard drive with a Protégé project extension (`.pprj`). This file appears as an option each time you log in to NCI Protégé. You can select it whenever you want to run the application locally.

Note: Working with a local project will not affect your ability to log in to the server and do your regular work. You can always close your local Protégé project and log in to the server database.

Perform the following procedures in the following order:

- *Downloading the Subset File* on page 33
- *Installing the Subset File* on page 33
- *Opening the Local Subset File in Protégé* on page 33
- *Setting Up a Local Project* on page 35

Downloading the Subset File

The *NCICB EVS Collaborative Terminology Development Tools* project on GForge includes a subset file of the NCI Thesaurus. You can use this file to set up your local project.

To obtain the local subset file from GForge, follow these steps:

1. Navigate to the following location on GForge:

   https://gforge.nci.nih.gov/projects/protegegui/

2. Click the **Files** tab.
3. Locate the **OWL File** category > **NCIT Subset 060926 w Retired** subcategory.
4. Select the file named `Thesaurus-ByName-060926-Retired.zip`.
5. Save the file to a directory on your hard drive, noting where you saved it.

Installing the Subset File

Prerequisite procedure: *Downloading the Subset File* on page 33

To install the local subset file downloaded from GForge, follow these steps:

1. Navigate to the directory where you downloaded the subset file.
2. (Optional) If you plan to store the file in a different location from your download directory, copy the file and paste it to the new location.

   Note: Do not copy the test file to the current Protégé application folder. Instead, create a folder for local Protégé files and projects, then copy the file there. This will keep your project files separate from Protégé application files.

3. Right-click the file name, then select the option to extract the file in the current folder or in another folder.

   The name of the extracted file is identical to the compressed file, but with an added `.owl` extension.

Opening the Local Subset File in Protégé

Prerequisite procedure: *Installing the Subset File* on page 33

After downloading and installing a local Thesaurus subset file on your hard drive, you can open the file in Protégé and associate it with a Protégé project.

To open a local subset file in Protégé, follow these steps:

1. In Windows Explorer, navigate to the following location:

   ```
   C:\Program Files\Protege.Client-1.2.x
   ```

   Note: The name of the folder includes the current version of NCI Protégé.

2. Double-click the following file: `run_protege.bat`.

   A Console (command line) window opens, followed by two other windows:

- The Protégé main application window, and
- The Welcome to Protégé window, displayed in front.

3. In the Welcome to Protégé window, select **Open Other** (*Figure 3.6*).

Open Other

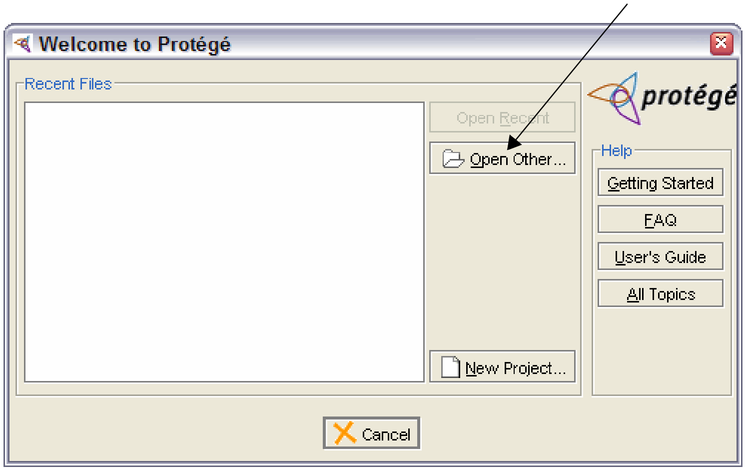
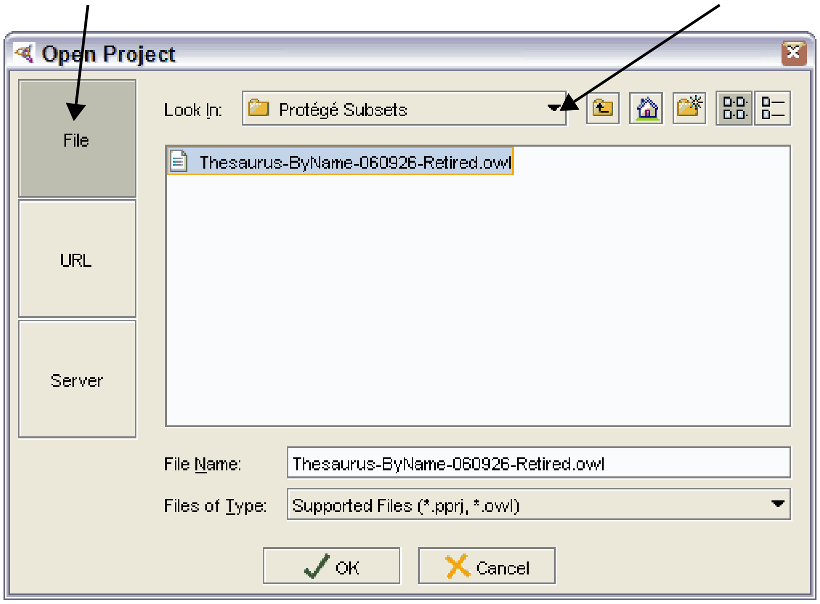


Figure 3.6 Welcome to Protégé window

4. In the Open Project window, select **File** (*Figure 3.7*).
5. Use the **Look In** list to browse for the file.

Select File...

...then browse to the file.



Look In: Protégé Subsets

Thesaurus-ByName-060926-Retired.owl

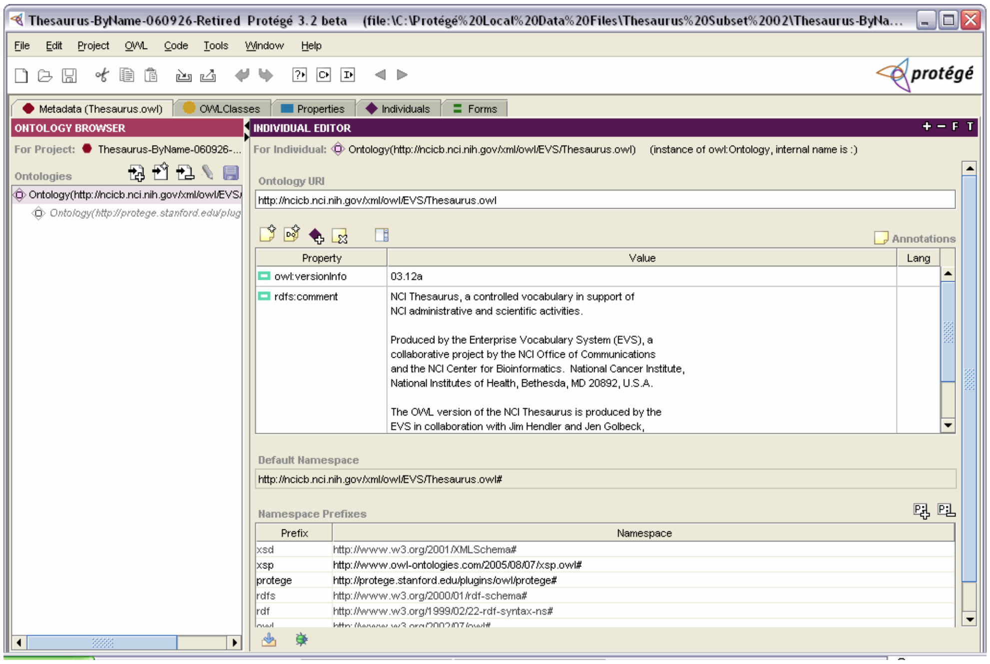
File Name: Thesaurus-ByName-060926-Retired.owl

Files of Type: Supported Files (*.pprj, *.owl)

Figure 3.7 Open Project window - File button

6. Select the file, then click **OK**.

NCI Protégé reads the file and then opens it. Note that the NCI-specific tabs are not yet visible. To configure your local project to match the server environment, see *Setting Up a Local Project* on page 35.



For Project: Thesaurus-ByName-060926-...

Ontology(http://ncicb.nci.nih.gov/xml/owl/EVS/

Ontology(http://protege.stanford.edu/plug

For Individual: Ontology(http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl) (instance of owl:Ontology, internal name is :)

Ontology URI: http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl

Property	Value	Lang
owl:versionInfo	03.12a	
rdfs:comment	NCI Thesaurus, a controlled vocabulary in support of NCI administrative and scientific activities. Produced by the Enterprise Vocabulary System (EVS), a collaborative project by the NCI Office of Communications and the NCI Center for Bioinformatics. National Cancer Institute, National Institutes of Health, Bethesda, MD 20892, U.S.A. The OWL version of the NCI Thesaurus is produced by the EVS in collaboration with Jim Hendler and Jen Golbeck,	

Default Namespace: http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#

Namespace Prefixes

Prefix	Namespace
xsd	http://www.w3.org/2001/XMLSchema#
xsp	http://www.owl-ontologies.com/2005/08/07/xsp.owl#
protege	http://protege.stanford.edu/plugins/owl/protege#
rdfs	http://www.w3.org/2000/01/rdf-schema#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#

Figure 3.8 Protégé standard interface with no visible NCI-specific tabs

Setting Up a Local Project

Prerequisite procedures:

- *Downloading the Subset File* on page 33
- *Installing the Subset File* on page 33
- *Opening the Local Subset File in Protégé* on page 33

When you first open a local Thesaurus subset file in NCI Protégé, the interface does not include the NCI plug-ins. This topic explains how to configure the local Protégé interface so that it mirrors the server environment.

To configure your local NCI Protégé environment, follow these steps:

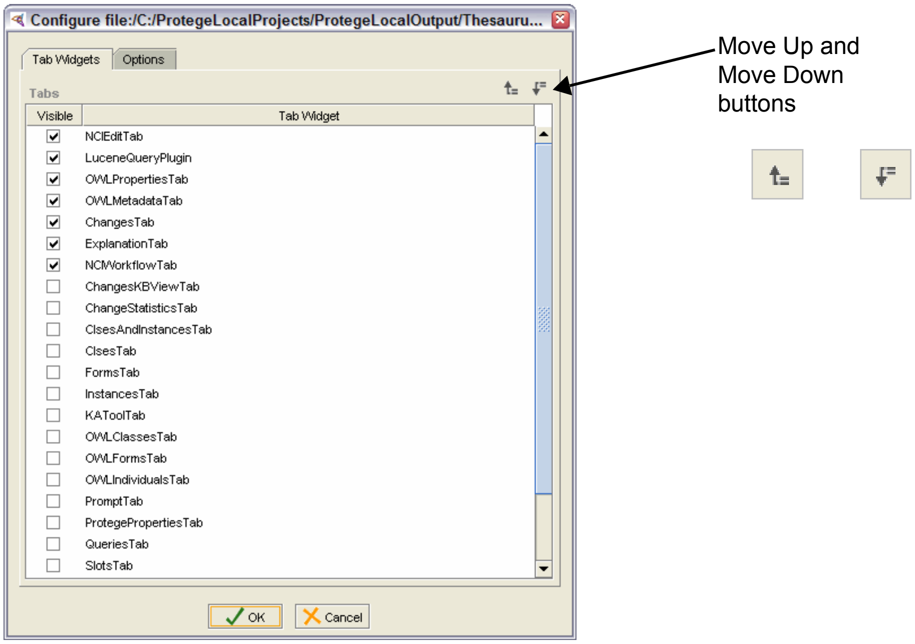
1. Select the following menu command: **Project** > **Configure...**

   The Configure File window opens.

2. Ensure that the **Tab Widgets** tab is selected.
3. In the **Visible** column, uncheck the boxes preceding each checked tab widget.
4. Check the box to the left of the **NCIEditTab** widget.
5. Repeatedly click the **Move Up** button to move the tab to the top of the list.

6. Repeat steps 4 and 5 for each of the following tab widgets, moving the widget to the specified position:
   - **LuceneQueryPlugin** (second position)
   - **OWLPropertiesTab** (third position)
   - **OWLMetadataTab** (fourth position)
   - **ChangesTab** (fifth position)
   - **ExplanationTab** (sixth position)
   - **NCIWorkflow Tab** (seventh position)

   The Configure file window should now resemble *Figure 3.9*.

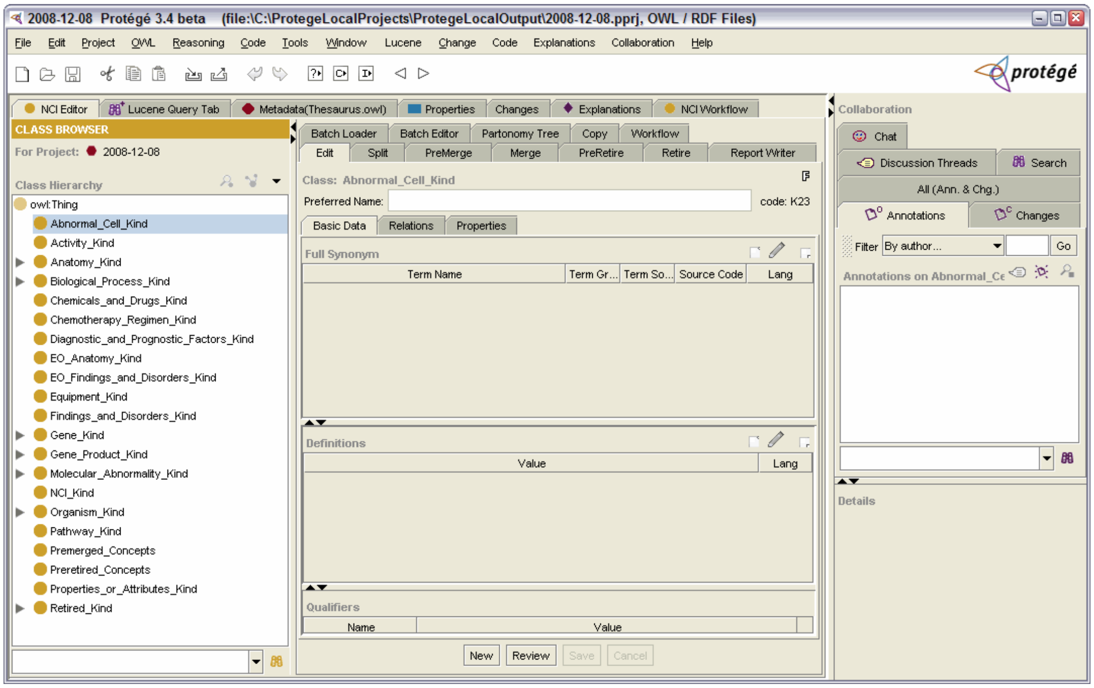


Visible	Tab Widget
☑	NCIEditTab
☑	LuceneQueryPlugin
☑	OWLPropertiesTab
☑	OWLMetadataTab
☑	ChangesTab
☑	ExplanationTab
☑	NCIWorkflowTab
☐	ChangesKBViewTab
☐	ChangeStatisticsTab
☐	ClsesAndInstancesTab
☐	ClsesTab
☐	FormsTab
☐	InstancesTab
☐	KAToolTab
☐	OWLClassesTab
☐	OWLFormsTab
☐	OWLIndividualsTab
☐	PromptTab
☐	ProtegePropertiesTab
☐	QueriesTab
☐	SlotsTab

Figure 3.9 Configure File window with ordered tab widgets

7. Click **OK** to close the window.

The tab display now matches the server configuration (*Figure 3.10*).



For Project: 2008-12-08

Class: Abnormal_Cell_Kind

code: K23

Filter By author...

Figure 3.10 New local configuration

8. Save the new project using either of the following menu commands:
   - Select **File** > **Save Project** to use the original file name as the project name.
   - Select **File** > **Save Project As** to use a new project name.

Tip: The next time you launch Protégé to run locally, your saved project will appear in the Welcome to Protégé window, under the Recently Accessed Files list. It will also appear in the Recent Files list under the File menu command (**File** > **Open Recent** > **[Project Name].pprj**).

CHAPTER
4
ABOUT THE NCI PROTÉGÉ WORK ENVIRONMENT

This chapter provides an overview of the NCI Protégé work environment.

Topics in this Chapter

About NCI Protégé

Protégé is an open-source development environment for ontologies and knowledge-based systems. The OWL plug-in extends the environment to support the Web Ontology Language (OWL).

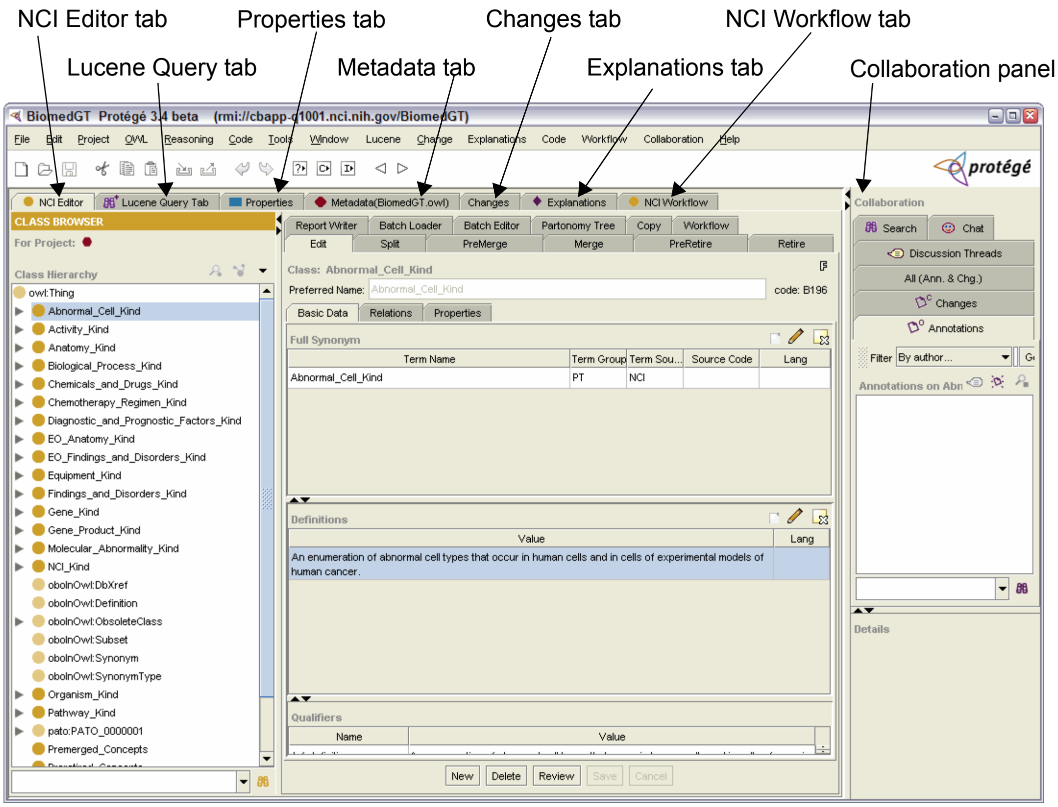
Developed at Stanford Medical Informatics, Protégé and the OWL plug-in support ontology editing in a multi-user, client-server environment. Using these tools, users at geographically dispersed locations can concurrently edit the same ontology data.

NCI Protégé is a collection of Protégé plug-ins developed for the NCI-specific editing environment. NCI Protégé provides a customized OWL editing environment tailored to the needs of NCI terminologies, including additional editing and querying capabilities that are not available in the Protégé OWL plug-in.

The NCI-specific plug-ins have been implemented as a tabbed interface. This chapter introduces the various interface elements. Other chapters provide more details about specific features in the context of the tasks that they are designed to perform.

About the Main Protégé Window

The main NCI Protégé window is organized in a series of tabs and panels, as shown in *Figure 4.1*.



Class: Abnormal_Cell_Kind

Preferred Name: Abnormal_Cell_Kind code: B196

Term Name	Term Group	Term Sou...	Source Code	Lang
Abnormal_Cell_Kind	PT	NCI		

Value	Lang
An enumeration of abnormal cell types that occur in human cells and in cells of experimental models of human cancer.	

Figure 4.1 Main NCI Protégé window

Table 4.1 describes the main tabs shown in *Figure 4.1*.

Feature	***Description***
NCI Editor tab	Provides access to a number of subtabs that represent various editing procedures. These include Edit, Split, PreMerge, Merge, PreRetire, Retire, Report Writer, Batch Loader, Batch Editor, Partonomy Tree, Copy, and Workflow. For more information, see *About the NCI Editor Tab* on page 42.
Lucene Query tab	Used for building both simple and advanced queries. From this tab, you can link to details for a selected search result. For more detailed information about searching and building advanced queries, see *Building an Advanced Query* on page 55.
Metadata tab	Used to examine metadata for the entire vocabulary. This includes such information as the creator, the version, and referenced external vocabularies. Only workflow managers can edit the metadata that is available on this tab.

Table 4.1 Description of main window features

Feature	***Description***
Properties tab	Used to look up the definition of any property in the database. Only workflow managers can edit the properties that are available on this tab.
Changes tab	Supports the Prompt plug-in by analyzing changes between baselines of the main ontology. Though this tab is used mostly by workflow managers, you can use it to trace a series of edit actions by examining the output that is shown in the top panel. To examine changes, you would need to manually step through them.
Explanations tab	Explains why a class was not properly classified by the reasoner. If you generate an inconsistent class, you can use this tab to find out why. For more information, see page 86.
NCI Workflow tab	Used by both workflow managers and editors (modelers) to manage work assignments. Managers use this tab to import packages (proposals) from the BiomedGT wiki and assign workflow items to modelers. Modelers use the tab to view and manage their assignments. For more information, see *Workflow for NCI Editors* on page 161.
Collaboration panel	Supports collaborative ontology editing. Using this panel, you can • Annotate classes, properties, and instances • Annotate changes such as creating or renaming classes • Participate in discussion threads • Initiate proposals and voting • Search and filter annotations using various criteria • Chat with other users who are connected to the NCI Protégé server The Collaboration panel remains visible regardless of which tab you select. To hide this panel, click the **Collapse** button ▶ on the top left edge of the panel. To show it again, click the **Expand** button ◀. **Note:** For more information about this panel, visit the following Stanford Web site: http://protege.stanford.edu/doc/collab-protege/

Table 4.1 Description of main window features (Continued)

About the NCI Editor Tab

When you first log in to Protégé, the NCI Editor tab is the prominent tab. This tab displays information about a selected class. *Figure 4.2* shows the tab layout.

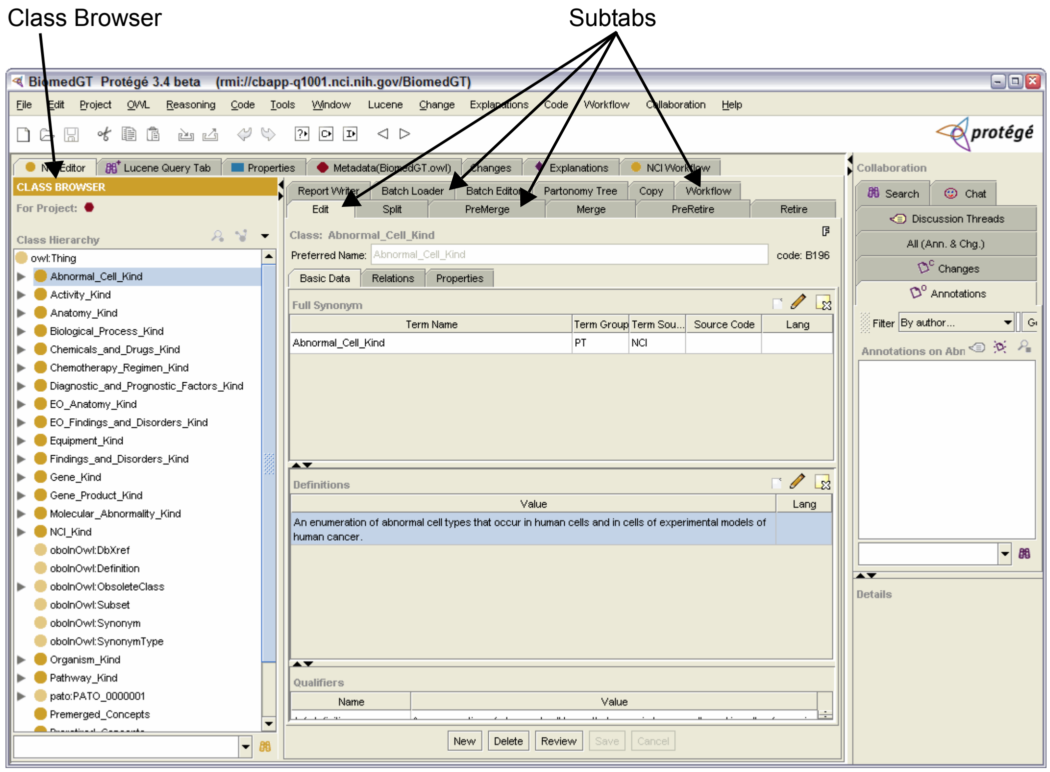


Figure 4.2 NCI Editor tab with Class Browser and subtabs

Table 4.2 describes the main features of the NCI Editor tab.

Feature	***Description***
Class Browser	Shows classes and subclasses in a tree view. You can select a class or subclass and view its details using the Edit subtab on the right. Below the tree, the Class Browser also includes a Search field for entering search terms and a button for executing the search. The results appear in a standalone Advanced Query window. For more detailed information, see *About the Class Browser* on page 43.
Subtabs	Enable you to perform various editing procedures. For a description of each subtab and cross-references to more detailed information, see *About the NCI Editor Subtabs* on page 44.

Table 4.2 NCI Editor tab: main features

About the Class Browser

Appearing on the left side of the Protégé main window, the Class Browser shows a tree view of all available superclasses and their subclasses (*Figure 4.3*).

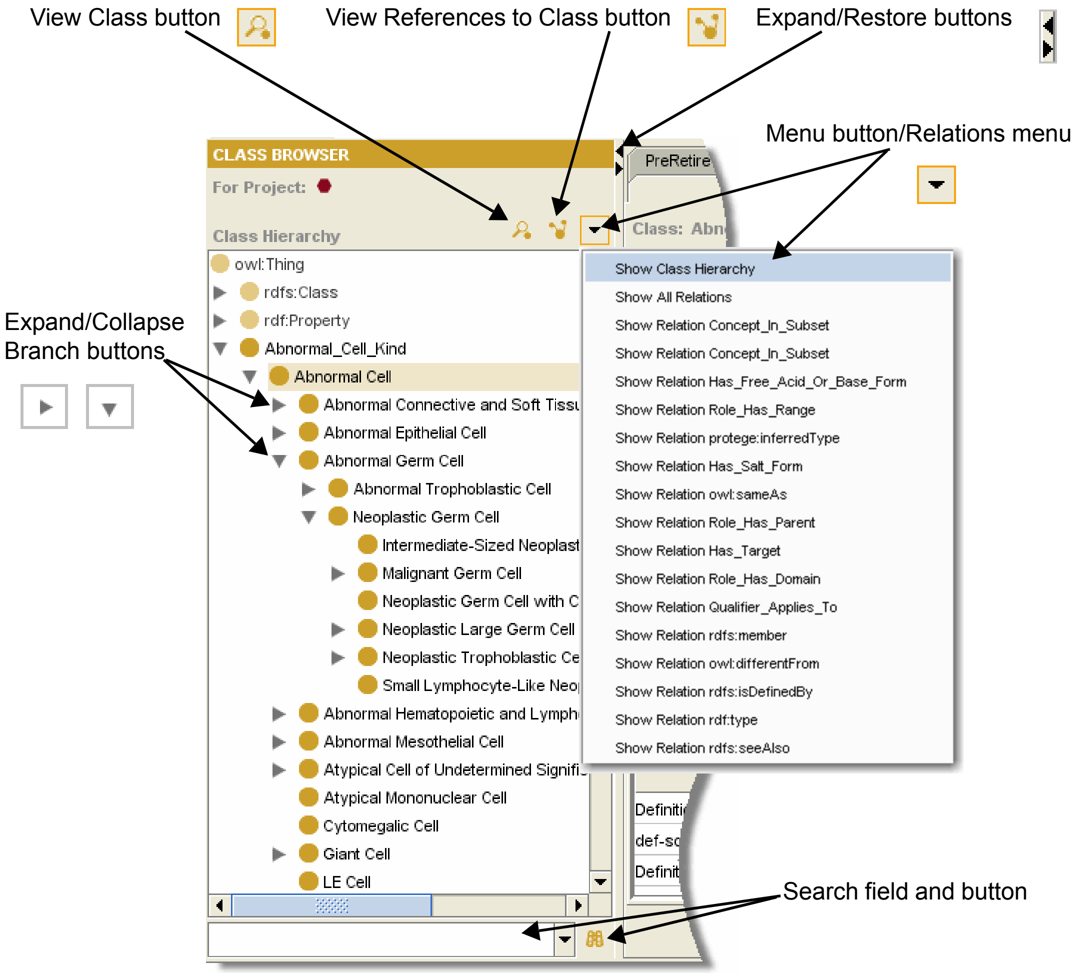


Figure 4.3 Class Browser

Table 4.3 describes the features illustrated in *Figure 4.3*.

Feature	***Description***
View Class button	Opens a read-only window showing basic data, relations, and properties for the selected class.
View References to Class button	Opens a read-only window showing references to the selected class.
Expand/Restore buttons	• Expand button: Expands the browser to fill the entire window. • Restore button: Restores the browser to the left panel.
Menu button/Relations menu	Shows the Relations menu, from which you can select a number of data views. To restore the view to show all classes, select the **Show Class Hierarchy** command.

Table 4.3 Class Browser features

Feature	**Description**
Expand/Collapse Branch buttons	Shows/hides subclasses of a selected class. ▶ ▼
Search field and button	Enable you to type a search string and execute a search.

Table 4.3 Class Browser features

About the NCI Editor Subtabs

Table 4.4 provides a brief description of each NCI Editor subtab. For more detailed information about a subtab, see the cross-referenced topic in the More Information column.

Subtab	**Description**	**More Information**
Edit	Displays annotation and relation data for a selected class. You can use the Basic, Relations, and Properties subtab to view and edit properties, restrictions, and associations.	See *Examining Class Information* on page 82.
Split	Creates a new class from an existing class.	See *Splitting a Class* on page 119.
PreMerge	Enables you to select and flag two classes for a merge. Although you can apply a pre-merge flag to classes, only workflow managers can perform an actual merge.	See *Merging Classes* on page 124.
Merge	(Workflow managers only) Enables a workflow manager to merge two classes that have been flagged using the pre-merge action.	See *Merging Classes* on page 124.
PreRetire	Enables you to flag a class for retirement. Although you can apply a pre-retire flag to a class, only workflow managers can retire a class.	See *Chapter 9, Pre-Retiring and Retiring Classes,* on page 133.
Retire	(Workflow managers only) Enables a workflow manager to retire a class that has been flagged by a pre-retire action.	See *Chapter 9, Pre-Retiring and Retiring Classes,* on page 133.
Report Writer	Generates reports for selected classes. You can save reports as external files and share them with colleagues.	See *Generating Reports for Selected Classes* on page 143.
Batch Loader	Enables you to load a batch of classes from an external file, save them to the Protégé database, and edit them.	See *Loading a Batch of Classes for Editing* on page 150.

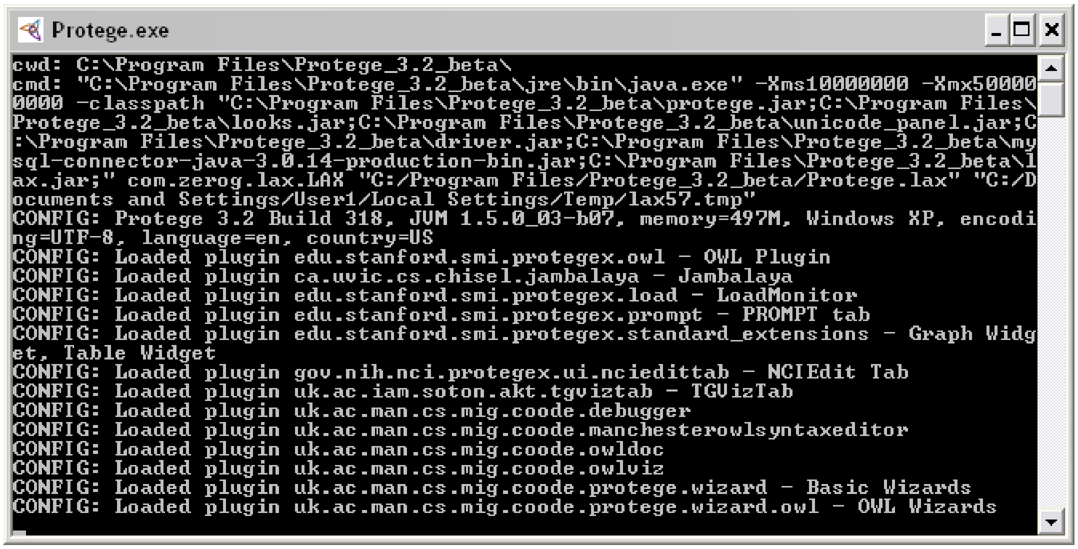
Table 4.4 Descriptions of NCI Editor subtabs

Subtab	**Description**	**More Information**
Batch Editor	Enables you to load a batch of classes that have been edited in an external application and save them to the Protégé database.	See *Loading a Batch of Edited Classes* on page 155.
Partonomy Tree	Shows classes that are connected by *part_of* relations. This tab enables you to select a root class and display it as a partonomy tree.	See *Viewing Classes Connected by part_of Relations* on page 96.
Copy	Clones a class from an existing class; also facilitates simultaneous editing of two classes.	See *Cloning a Class* on page 130.
Workflow	Used to examine concepts related to incoming proposals from the BiomedGT wiki.	See *Workflow for NCI Editors* on page 161.

Table 4.4 Descriptions of NCI Editor subtabs

Working with the Console

When you launch NCI Protégé, a command line window called the *Console* opens, followed by other windows that are a part of the familiar graphical user interface (GUI). The Console (*Figure 4.4*) remains open in the background each time you run a Protégé session.



```
cwd: C:\Program Files\Protege_3.2_beta\
cmd: "C:\Program Files\Protege_3.2_beta\jre\bin\java.exe" -Xms10000000 -Xmx50000
0000 -classpath "C:\Program Files\Protege_3.2_beta\protege.jar;C:\Program Files\
Protege_3.2_beta\looks.jar;C:\Program Files\Protege_3.2_beta\unicode_panel.jar;C
:\Program Files\Protege_3.2_beta\driver.jar;C:\Program Files\Protege_3.2_beta\my
sql-connector-java-3.0.14-production-bin.jar;C:\Program Files\Protege_3.2_beta\l
ax.jar;" com.zerog.lax.LAX "C:/Program Files/Protege_3.2_beta/Protege.lax" "C:/D
ocuments and Settings/User1/Local Settings/Temp/lax57.tmp"
CONFIG: Protege 3.2 Build 318, JVM 1.5.0_03-b07, memory=497M, Windows XP, encodi
ng=UTF-8, language=en, country=US
CONFIG: Loaded plugin edu.stanford.smi.protegex.owl - OWL Plugin
CONFIG: Loaded plugin ca.uvic.cs.chisel.jambalaya - Jambalaya
CONFIG: Loaded plugin edu.stanford.smi.protegex.load - LoadMonitor
CONFIG: Loaded plugin edu.stanford.smi.protegex.prompt - PROMPT tab
CONFIG: Loaded plugin edu.stanford.smi.protegex.standard_extensions - Graph Widg
et, Table Widget
CONFIG: Loaded plugin gov.nih.nci.protegex.ui.ncieditab - NCIEdit Tab
CONFIG: Loaded plugin uk.ac.iam.soton.akt.tgviztab - TGVizTab
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.debugger
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.manchesterowlsyntaxeditor
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.owldoc
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.owlviz
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.protege.wizard - Basic Wizards
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.protege.wizard.owl - OWL Wizards
```

Figure 4.4 Console window

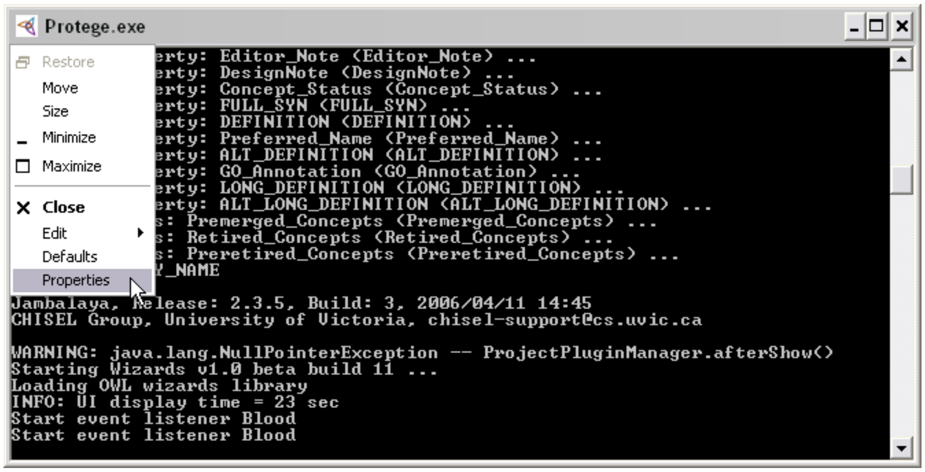
The Console displays a line-by-line record of the actions that you perform while working with NCI Protégé. It also provides a snapshot of Protégé's state at any given time. The Development team can use this information to troubleshoot any problems that you may have.

Setting the Console Display Capacity

Note: This procedure increases the amount of information that the Console can display, enabling you to provide more information for the Development team. Be sure to complete this procedure before performing the following two procedures.

The Console shows around 300 lines of text. This value limits the amount of information that can be captured for troubleshooting purposes. To increase the window's display capacity, follow these steps:

1. Access the Console menu by clicking the icon in the upper left corner of the window, or by right-clicking the blank area of the window title bar.
2. Select the **Properties** command (*Figure 4.5*).

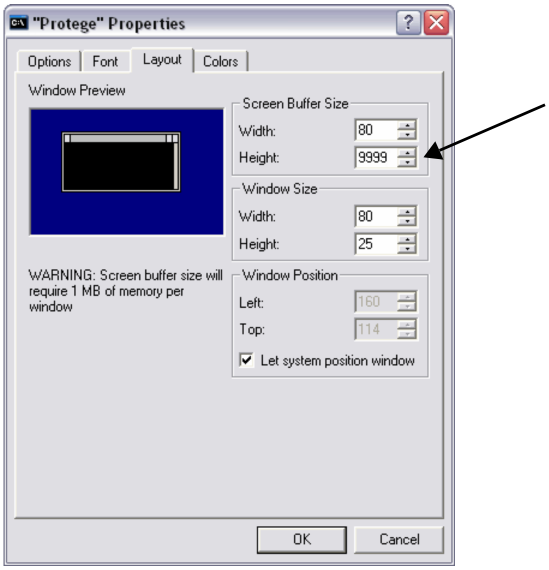


```
erty: Editor_Note (Editor_Note) ...
erty: DesignNote (DesignNote) ...
erty: Concept_Status (Concept_Status) ...
erty: FULL_SYN (FULL_SYN) ...
erty: DEFINITION (DEFINITION) ...
erty: Preferred_Name (Preferred_Name) ...
erty: ALT_DEFINITION (ALT_DEFINITION) ...
erty: GO_Annotation (GO_Annotation) ...
erty: LONG_DEFINITION (LONG_DEFINITION) ...
erty: ALT_LONG_DEFINITION (ALT_LONG_DEFINITION) ...
s: Premerged_Concepts (Premerged_Concepts) ...
s: Retired_Concepts (Retired_Concepts) ...
s: Preretired_Concepts (Preretired_Concepts) ...
Y_NAME
Jambalaya, Release: 2.3.5, Build: 3, 2006/04/11 14:45
CHISEL Group, University of Victoria, chisel-support@cs.uvic.ca

WARNING: java.lang.NullPointerException -- ProjectPluginManager.afterShow()
Starting Wizards v1.0 beta build 11 ...
Loading OWL wizards library
INFO: UI display time = 23 sec
Start event listener Blood
Start event listener Blood
```

Figure 4.5 Console window menu - Properties command

The Properties window opens (*Figure 4.6*).

3. In the **Layout** tab > **Screen Buffer Size** section, type `9999` in the **Height** field.



Screen Buffer Size
Width: 80
Height: 9999

Window Size
Width: 80
Height: 25

Window Position
Left: 160
Top: 114
☑ Let system position window

WARNING: Screen buffer size will require 1 MB of memory per window

Screen Buffer Size - Height value

Figure 4.6 Console Properties window

This sets the value to a maximum so that you will not lose useful information.

4. Click **OK** to close the Console Properties window.
5. If the Apply Properties to Shortcut window opens, select the default: **Apply properties to current window only**.

Capturing Information for Troubleshooting

Prerequisite procedure: *Setting the Console Display Capacity* on page 46

The following topics explain how to capture Console information and submit the information as a file attachment when reporting a problem.

- *Capturing Information and Copying it from the Console Window* on this page
- *Submitting a Bug Report and File Attachment to GForge* on page 48

Capturing Information and Copying it from the Console Window

A *thread dump* captures information stored in memory and appends it to the information displayed in the Console. You can copy all of the displayed window content, paste it into an external text file, and post the file to the GForge Tracker.

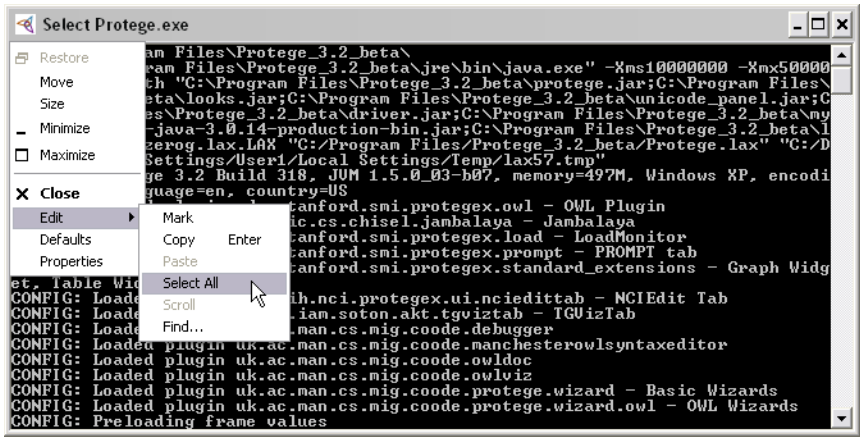
To capture and copy Console information, follow these steps:

1. Press CTRL + BREAK (or PAUSE BREAK, depending on your keyboard).

   The thread dump is appended to the content in the Console window.

2. Access the Console menu by
   - clicking the icon in the upper left corner of the window, or
   - right-clicking the blank area of the window title bar.

3. Select **Edit** > **Select All** to select all of the text in the window.



```
              am Files\Protege_3.2_beta\
              ram Files\Protege_3.2_beta\jre\bin\java.exe" -Xms10000000 -Xmx50000
              th "C:\Program Files\Protege_3.2_beta\protege.jar;C:\Program Files\
              eta\looks.jar;C:\Program Files\Protege_3.2_beta\unicode_panel.jar;C
              es\Protege_3.2_beta\driver.jar;C:\Program Files\Protege_3.2_beta\my
              -java-3.0.14-production-bin.jar;C:\Program Files\Protege_3.2_beta\l
              zerog.lax.LAX "C:/Program Files/Protege_3.2_beta/Protege.lax" "C:/D
              Settings/User1/Local Settings/Temp/lax57.tmp"
              ge 3.2 Build 318, JVM 1.5.0_03-b07, memory=497M, Windows XP, encodi
              guage=en, country=US
                           tanford.smi.protegex.owl - OWL Plugin
                           ic.cs.chisel.jambalaya - Jambalaya
                           tanford.smi.protegex.load - LoadMonitor
                           tanford.smi.protegex.prompt - PROMPT tab
                           tanford.smi.protegex.standard_extensions - Graph Widg
et, Table Wid
CONFIG: Loade              ih.nci.protegex.ui.nciedittab - NCIEdit Tab
CONFIG: Loade              .iam.soton.akt.tgviztab - TGVizTab
CONFIG: Loade              .man.cs.mig.coode.debugger
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.manchesterowlsyntaxeditor
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.owldoc
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.owlviz
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.protege.wizard - Basic Wizards
CONFIG: Loaded plugin uk.ac.man.cs.mig.coode.protege.wizard.owl - OWL Wizards
CONFIG: Preloading frame values
```

Figure 4.7 Command Line window menu - Select All command

4. Access the Console menu again, then select **Edit** > **Copy**.

5. Using a text editor such as Windows Notepad, paste the copied text into a blank text file.
6. Save the file with a name such as `console-copy_<date>.txt`, where *<date>* is today’s date in MM-DD-YYYY format (for example, `console-copy_02-12-08`.txt).

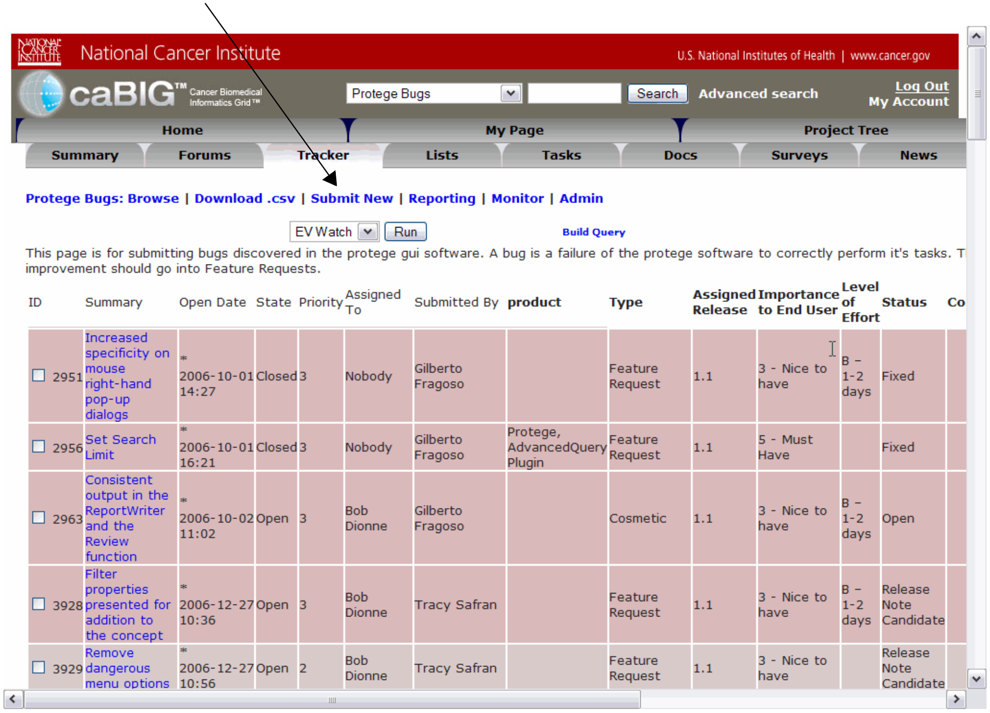
Submitting a Bug Report and File Attachment to GForge

While working with Protégé, you may sometimes want to report a problem to the development team. You can use the Tracker tab on GForge to report a bug.

To report a bug using the GForge Bug Tracker, follow these steps:

1. Using your browser, go to https://gforge.nci.nih.gov/.
2. Enter your username and password.
3. Click the **My Page** tab.
4. Under **My Projects**, select **EVS Collaborative Terminology Dev Tools**.
5. Click the **Tracker** tab.
6. In the **Tracker** column on the left, select **Protege Bugs**.
7. In the top row of links, select **Submit New** (*Figure 4.8*).

Select Submit New



This page is for submitting bugs discovered in the protege gui software. A bug is a failure of the protege software to correctly perform it's tasks. T improvement should go into Feature Requests.

	ID	Summary	Open Date	State	Priority	Assigned To	Submitted By	product	Type	Assigned Release	Importance to End User	Level of Effort	Status	Co
☐	2951	Increased specificity on mouse right-hand pop-up dialogs	* 2006-10-01 14:27	Closed	3	Nobody	Gilberto Fragoso		Feature Request	1.1	3 - Nice to have	B - 1-2 days	Fixed	
☐	2956	Set Search Limit	* 2006-10-01 16:21	Closed	3	Nobody	Gilberto Fragoso	Protege, AdvancedQuery Plugin	Feature Request	1.1	5 - Must Have		Fixed	
☐	2963	Consistent output in the ReportWriter and the Review function	* 2006-10-02 11:02	Open	3	Bob Dionne	Gilberto Fragoso		Cosmetic	1.1	3 - Nice to have	B - 1-2 days	Open	
☐	3928	Filter properties presented for addition to the concept	* 2006-12-27 10:36	Open	3	Bob Dionne	Tracy Safran		Feature Request	1.1	3 - Nice to have	B - 1-2 days	Release Note Candidate	
☐	3929	Remove dangerous menu options	* 2006-12-27 10:56	Open	2	Bob Dionne	Tracy Safran		Feature Request	1.1	3 - Nice to have		Release Note Candidate	

Figure 4.8 Tracker page - Submit New link

8. Complete the fields listed in *Table 4.5*. Fields are listed across, and then down.

Field	***Description***
Status	Leave as is. The Development team assigns the status indicator.
Component	Select the application component to which the problem applies.
Level of Effort	Do not use this field. It will eventually be removed from the form.
Importance to End User	Select a priority level.
Assigned Release	Select the appropriate release number from the list.
Type	Select a description for the nature of the problem.
Product	Select the name of the product in which the defect was found.
Level of Difficulty	Leave as is. The developer assigned to fix the bug sets the value.
Transition	Leave as is. The developer assigned to fix the bug sets this value.
Version Found	Enter the version number of the application that always reproduces the problem. The QA team uses this field to track the defect, so it must always show the correct version.
Previous Build Tested	Leave as is. The QA team uses it to show the version number of the most recent application version used to test the problem. Testers change this number whenever they verify a developer's fix. If the status of a problem is *Fixed*, this field tells you the exact version that contains the fix.
Level of Effort - High (in Days)	Leave as is. The Development team uses it to show the maximum estimated level of effort to fix the defect (code and unit test) in working days (assumes 100% resources working eight-hour days).
Level of Effort - Low (in Days) / Iteration	Leave as is. The Development team uses it to show the minimum estimated level of effort to fix the defect (code and unit test) in working days (assumes 100% resources working eight-hour days).
Assigned To	Leave as is. The Development team uses it to assign the request to a team member.
Priority	Leave as is. The Development uses it to assign a priority level between 1 (lowest) and 5 (highest).
Summary	Add a summary title for the problem.
Detailed Description (with Notepad icon)	Add a detailed description of the problem, including the procedure were you performing when the problem happened. If you were able to capture the text of any messages that appeared, include them here.

Table 4.5 Bug Tracker fields

Field	***Description***
Check to Upload & Attach File	1. Check the box if you are including a file. 2. Click **Browse**, then find the file that you created in *Capturing Information and Copying it from the Console Window* on page 47. 3. Upload the file.
File Description	Enter a brief description of the file that you included.

Table 4.5 Bug Tracker fields (Continued)

9. Click **Submit**.

   The message *Item Successfully Created* appears at the top of the page.

CHAPTER 5

SEARCHING

Protégé's fine-tuned search features enable you to quickly find classes without having to browse the entire Class Browser hierarchy. This chapter introduces you to both the simple and advanced search features available in NCI Protégé.

Topics in this Chapter

Simple vs. Advanced Search

The NCI Protégé search tools are similar to search tools used on the Web:

- You perform a *simple search* by typing a search string in a text field and then clicking a button to execute the search (*Figure 5.1*). By default, Protégé searches using the *Preferred Name* with exact matching.

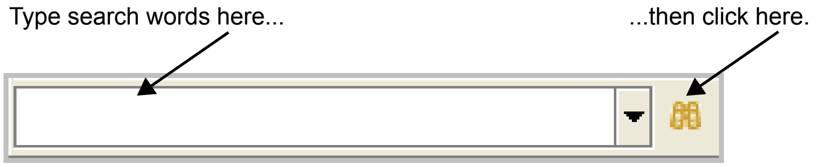
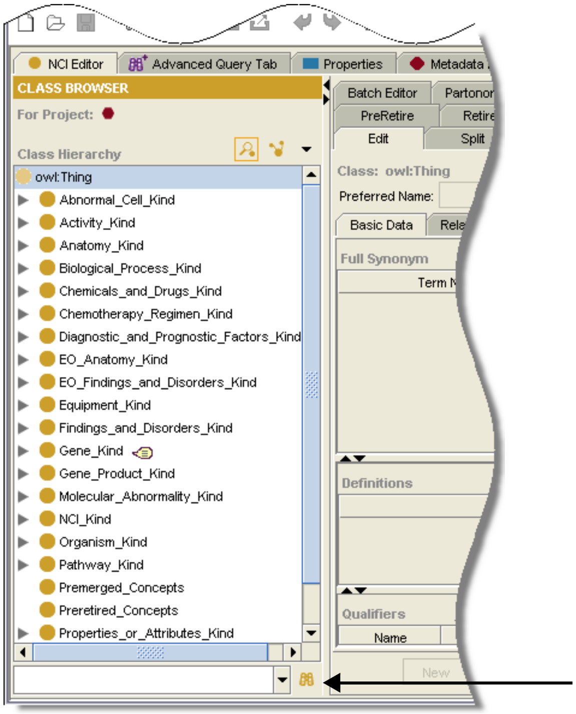


Figure 5.1 Simple search field and button

- You perform an *advanced search* by using the query-building features available on the Lucene Query tab.

Performing a Simple Search

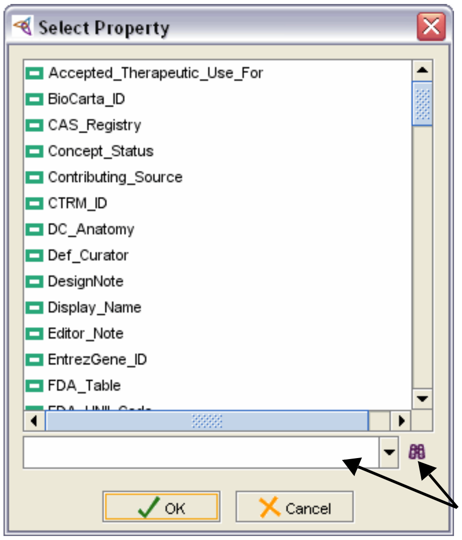
The Simple Search field and search button are available in multiple areas of the NCI Protégé interface. When you first launch the application, you can find the field and button on the NCI Editor tab, just below the Class Browser (*Figure 5.2*).



Simple Search field and button

Figure 5.2 NCI Editor tab with simple search field and button

Many standalone windows also include the Simple Search field and button. Instead of scrolling through long lists, you can search for a specific item in the displayed list. The field and button are located below the list, in the bottom of the window (*Figure 5.3*).



Simple Search field and button

Figure 5.3 Select Property window with simple search field and button

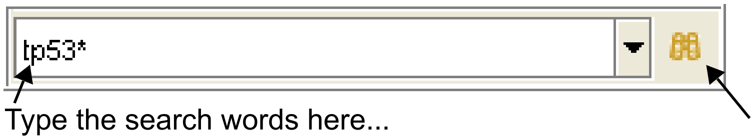
Using the Simple Search Field and Button

To perform a simple search, follow these steps:

1. Type all or part of a term in the **Simple Search** field.

   Tip: Searches are not case-sensitive. The example shown in *Figure 5.4* uses `tp53*`, but `TP53*` would return the same results.

2. Click the **Search** button to the right of the field.



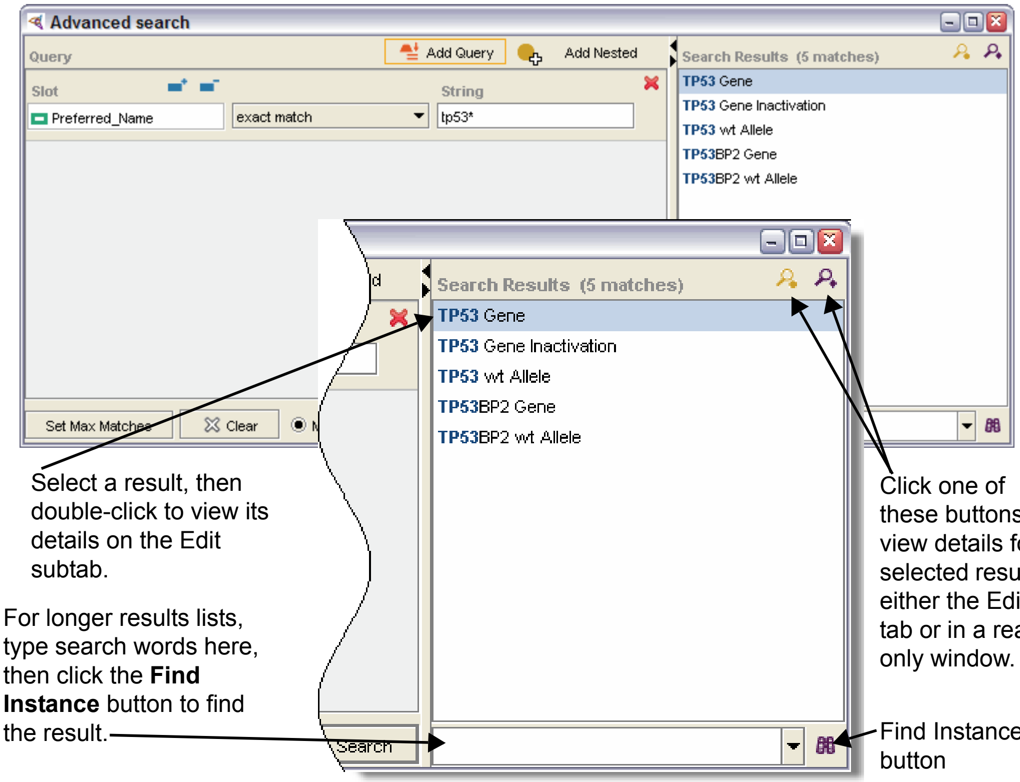
tp53*

Type the search words here...

...then click here.

Figure 5.4 Simple search field and Search for Class button

The Advanced search window opens (*Figure 5.5*). The layout of this window is identical to the layout of the Lucene Query tab, discussed in *Building an Advanced Query* on page 55. For simple searches, use the features in the right side of the window to select, search for, and display results.



Preferred_Name | exact match | tp53*

Search Results (5 matches)

- TP53 Gene
- TP53 Gene Inactivation
- TP53 wt Allele
- TP53BP2 Gene
- TP53BP2 wt Allele

Select a result, then double-click to view its details on the Edit subtab.

For longer results lists, type search words here, then click the **Find Instance** button to find the result.

Click one of these buttons to view details for a selected result in either the Edit tab or in a read-only window.

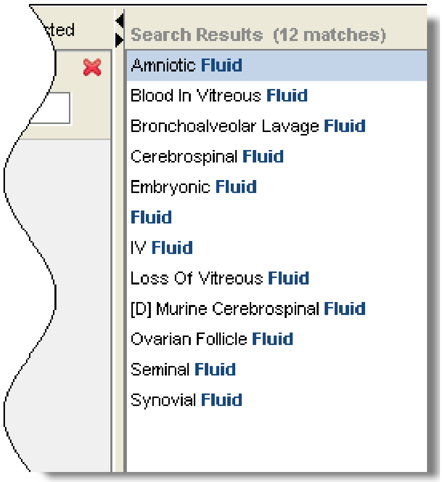
Find Instance button

Figure 5.5 Results for simple search

About the Search Results Display

NCI Protégé displays search results in alphabetical order, with each result represented by its preferred name. In the Search Results list, matching search strings are emphasized in bold text.

For example, *Figure 5.6* shows the results for the query `Preferred_Name ends with fluid`. Note that each instance of the word *fluid* shows in bold.



Search Results (12 matches)
Amniotic **Fluid**
Blood In Vitreous **Fluid**
Bronchoalveolar Lavage **Fluid**
Cerebrospinal **Fluid**
Embryonic **Fluid**
Fluid
IV **Fluid**
Loss Of Vitreous **Fluid**
[D] Murine Cerebrospinal **Fluid**
Ovarian Follicle **Fluid**
Seminal **Fluid**
Synovial **Fluid**

Figure 5.6 Results: Preferred_Name ends with fluid

Note also in *Figure 5.6* that the result *Murine Cerebrospinal Fluid* is preceded by the letter D enclosed in brackets: [D]. This indicates that the class is deprecated.

Selecting and Reviewing Search Results

If your search returns only one result, that result is automatically selected. If the search returns multiple results, follow these steps to select a result and view its details:

1. (Optional) To search through a long results list, follow these steps:
    a. Type a search word or phrase in the **Search** field below the list (*Figure 5.5* on page 53).
    b. Click the **Find Instance** button .
2. View details for a result using any of the following methods:
    - Select the result, then click the **View Instance** button to view the basic data, relations, and properties for the selected result in a standalone, read-only window. The window shows in the foreground.
    - Select the result, then click the **Edit Class** button to view the basic data, relations, and properties for the selected result on the **NCI Editor** tab > **Edit** subtab.
    - Double-click the selected result to view its basic data, relations, and properties on the **NCI Editor** tab > **Edit** subtab.

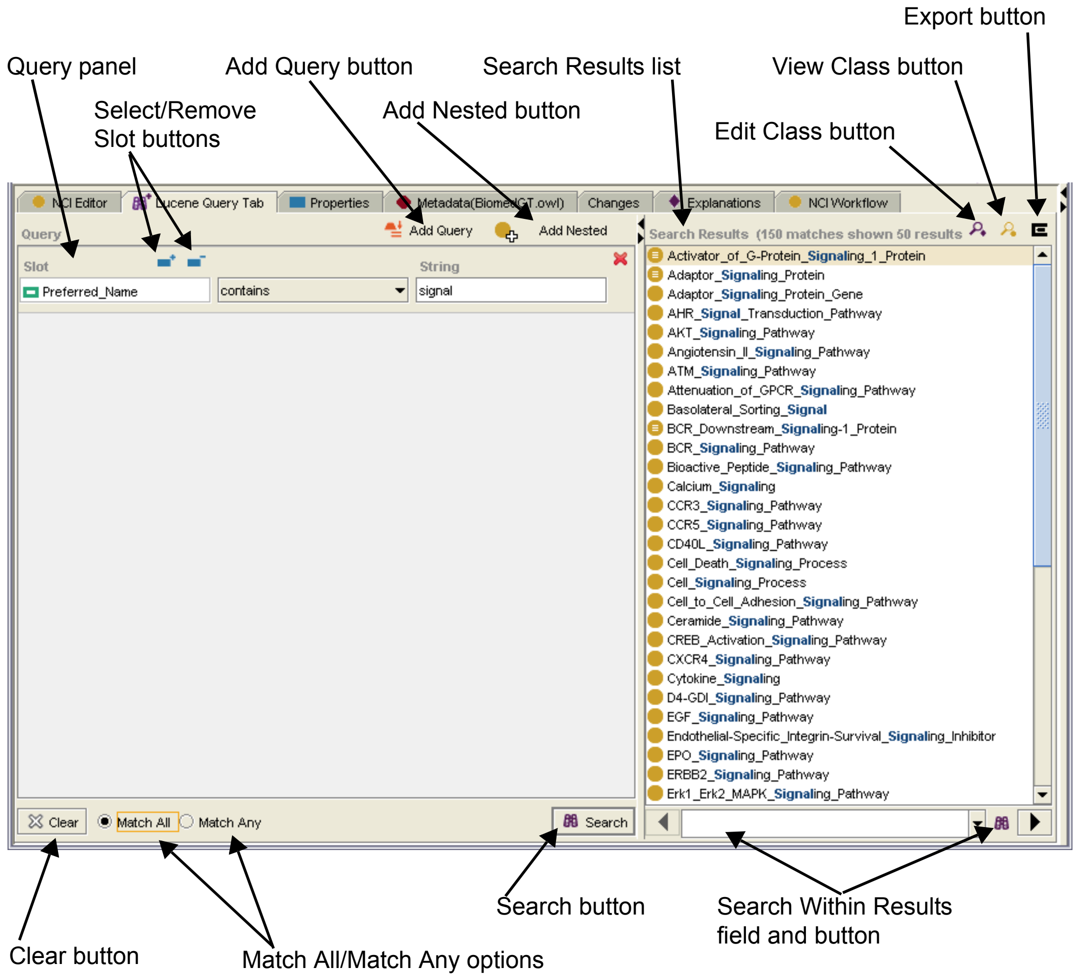
Note: The Advanced search window remains open unless you forcibly close it. You may need to minimize or close the window to view details for a selected result.

Building an Advanced Query

The Lucene Query tab gives you more control over the scope and complexity of searches. Using an advanced query, you can

- Search using a class name or any property, including a FULL_SYN, definition, or preferred name
- Search for classes that have specific restrictions
- Build search queries using several search parameters, such as *contains*, *starts with*, or *ends with*
- Narrow or broaden the search scope by selecting a Match All (AND) or Match Any (OR) option.

Figure 5.7 shows the layout of the Lucene Query tab. For an explanation of the main features, see *Table 5.1* (page 56).



Slot: Preferred_Name | contains | String: signal

● Match All ○ Match Any

Search Results (150 matches shown 50 results)

Activator_of_G-Protein_Signaling_1_Protein
Adaptor_Signaling_Protein
Adaptor_Signaling_Protein_Gene
AHR_Signal_Transduction_Pathway
AKT_Signaling_Pathway
Angiotensin_II_Signaling_Pathway
ATM_Signaling_Pathway
Attenuation_of_GPCR_Signaling_Pathway
Basolateral_Sorting_Signal
BCR_Downstream_Signaling-1_Protein
BCR_Signaling_Pathway
Bioactive_Peptide_Signaling_Pathway
Calcium_Signaling
CCR3_Signaling_Pathway
CCR5_Signaling_Pathway
CD40L_Signaling_Pathway
Cell_Death_Signaling_Process
Cell_Signaling_Process
Cell_to_Cell_Adhesion_Signaling_Pathway
Ceramide_Signaling_Pathway
CREB_Activation_Signaling_Pathway
CXCR4_Signaling_Pathway
Cytokine_Signaling
D4-GDI_Signaling_Pathway
EGF_Signaling_Pathway
Endothelial-Specific_Integrin-Survival_Signaling_Inhibitor
EPO_Signaling_Pathway
ERBB2_Signaling_Pathway
Erk1_Erk2_MAPK_Signaling_Pathway

Figure 5.7 Lucene Query tab

Table 5.1 lists and describes the main features of the Lucene Query tab.

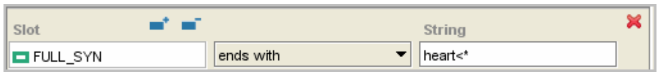
Feature	***Description***
Query panel	Shows a full search expression: slot, parameters (*e.g.*, *contains*, *starts with*) and a typed string.  Slot: FULL_SYN ends with String: heart<*
Select Slot button	Opens a window in which you can select a new slot (property) to replace the current slot.
Remove Slot button	Clears the Slot field, clears the parameters list, and removes the String field. To restore these features and add new query information, click the **Select Slot** button.
Add Query button  	Adds a new query, including a Slot field (with *Preferred_Name* displayed), a parameters list, and an empty String field. Use these tools to build your own expressions.
Add Nested button  	Enables you to build a query for finding classes with specific restrictions or associations.
Search Results list	Lists the returned results from a search and enables you to search within the list. See *Figure 5.7*.
Edit Class button	Shows the basic data, relations, and properties of a selected search result in the NCI Editor tab > Edit subtab.
View Class button	Opens a read-only window in which you can view basic data, relations, and properties for a selected search result.
Clear button  	Removes all expressions and leaves only one generic expression: *Preferred_Name contains <blank>*. This button does not affect the Search Results pane on the right.
Match All / Match Any options  ● Match All ○ Match Any	Enable you to control the scope of a search: • Match All (the default) performs an AND search. • Match Any performs an OR search.
Search button  	Executes a search.
Search Within Results field and button	Enable you to search for an item from the returned results list. Type a term in the field and click the button on the right to execute the search. 

Table 5.1 Lucene Query tab features

Advanced Query Syntax

When building queries, you can use simple or complex properties to search for classes. For example, you can use a simple property such as a preferred name, or you can use a complex property such as a FULL_SYN.

Note, however, that search strings for simple and complex properties are constructed differently. Because the underlying data for FULL_SYNs is described with XML tags, search strings used with complex properties require the opening (<) or closing (>) angle brackets that enclose the XML tags.

For many parameters, angle brackets are used in conjunction with an asterisk symbol (*). Although you can omit these characters in your queries, you will return more accurate results by including them.

Following are examples of how you can use angle brackets with asterisks to refine search results:

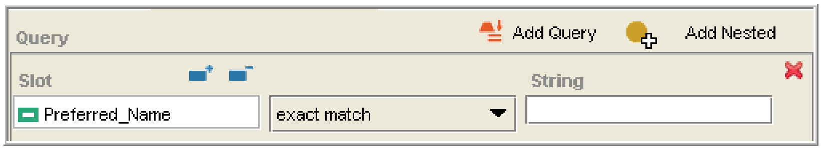
```
contains >heart<

starts with *>heart

ends with heart<*

exact match *>heart<*
```

To include angle brackets in search strings, press SHIFT + < (the less than symbol) for an opening bracket, and SHIFT + > (the greater than symbol) for a closing bracket.

Tips for Using the Query-Building Interface

This section highlights many useful query-building features on the Lucene Query tab.

- Build queries in the left pane of the tab. The default query uses *Preferred Name* as the default property and *exact match* as the default parameter. You supply the contents of the String field.



Slot: Preferred_Name exact match

Figure 5.8 Default query

- The Slot field shows the property used in the query. (*Slot* is a synonym for a property.)
    - To change the slot, click the **Select Slot** button. This opens the Select Slot window, in which you can select or search for a slot to replace the current one.
    - Click the **Remove Slot** button to clear the Slot field and the parameter list.

- When building a query expression, select one of the five parameters from the parameter list.

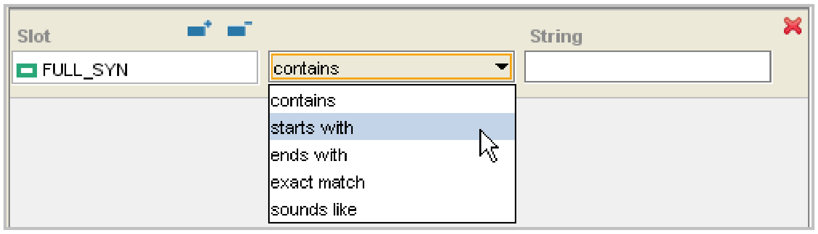


Figure 5.9 Parameter list

- Type a search string in the **String** field.
- Click the **Add Query** button to add quieries .
- Click the **Add Nested** button to add a nested query when searching for restrictions or associations. For more information, see *Building a Restriction-Based Query* on page 65.
- Click the **Remove Query** button to remove a single query. If you click this button when you already have only a single query, Protégé restores the default query (*Preferred_Name* with exact matching).
- Click the **Clear** button to remove all queries. After clearing the queries, Protégé displays the default query shown in *Figure 5.8*.
- The default search scope is Match All, which uses a Boolean AND. To test the results of your queries, try changing the scope option to Match Any, which uses a Boolean OR.

Advanced Query Examples

The following topics explain how to build various types of advanced queries:

- *Building a Simple Property Query: Preferred Name* on page 59
- *Building a Complex Property Query: FULL_SYN* on page 60
- *Building a Combined Property Query: Preferred_Name and FULL_SYN* on page 62
- *Building a Restriction-Based Query* on page 65
- *Building an Association-Based Query* on page 68

Building a Simple Property Query: Preferred Name

When you first access the Lucene Query tab, the left pane shows the default query (*Preferred_Name* with exact matching*)*. The following retains the default slot of Preferred_Name but uses a different parameter.

Scenario: You want to search for a class with the preferred name *Cerebrospinal Fluid*. You are unsure of the spelling of *cerebrospinal*, so you will use only the word *fluid*.

Procedure: To build and execute a simple property query, follow these steps:

1. Click the **Lucene Query** tab.

   As shown in *Figure 5.10*, the Query pane on the left shows the default query.

   Figure 5.10 Default query

2. Leave the **Slot** value as **Preferred_Name**.
3. Select **ends with** from the parameter list.
4. Type `fluid` in the **String** field.

   Note: If you make a mistake and want to clear the query, click the **Remove Query** button ✖, or click the **Clear** button ☒ Clear. This clears the **String** field and restores the default query.

5. Leave the search scope set to the default of **Match All** (Boolean AND).
6. Click the **Search** button Search.
7. In the **Search Results** pane on the right, double-click **Cerebrospinal Fluid***.*

   Protégé switches to the **NCI Editor** > **Edit** subtab and displays FULL_SYNs, definitions, and qualifiers for the selected class.

Building a Complex Property Query: FULL_SYN

Since the following query uses FULL_SYN as the slot, it requires an opening (<) or closing (>) angle bracket (or possibly both brackets) used to enclose the underlying XML tags.

Scenario: You want to search for a class using the FULL_SYN *Benign Breast Neoplasm*. Following is the underlying XML construction:

```
<term-name>Benign Tumor of Breast</term-name><term-group>SY
</term-group><term-source>NCI</term-source>
```

You could search for this class using either of two queries:

- `FULL_SYN starts with *>benign`
- `FULL_SYN ends with breast<*`

Procedure: To build and execute a complex property query, follow these steps:

1. Click the **Lucene Query** tab.

   As shown in *Figure 5.11*, tThe Query pane on the left shows the default query.

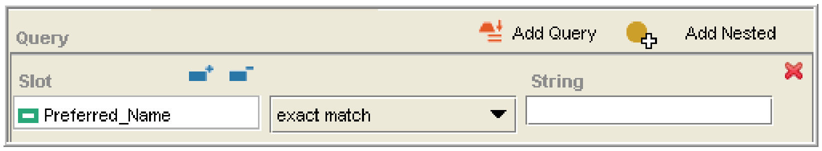


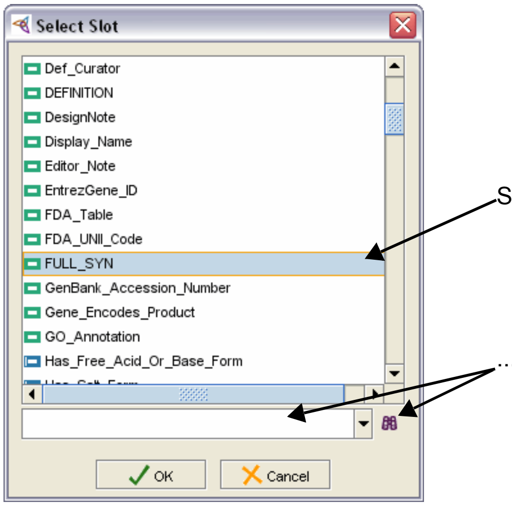
Figure 5.11 Default query

   You now need to replace the default property with a FULL_SYN property.

2. Click the **Select Slot** button .

   The Select Slot window opens (*Figure 5.12*).

3. Scroll until you see FULL_SYN property, or search for it.

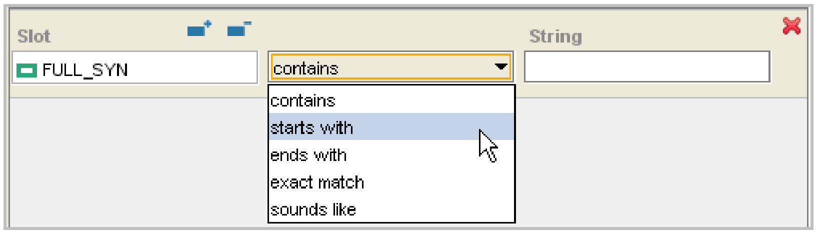


Scroll to the item in the list...

...or enter a term and click the **Find** button.

Figure 5.12 Select Slot window

4. Click **OK** to close the Select Slot window.
5. Select **starts with** from the parameter list.

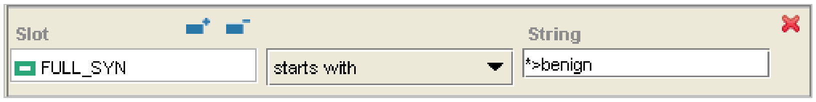


Slot: FULL_SYN
contains
String:

Figure 5.13 Parameter list - starts with

6. In the **String** field, type `*>benign`.

   The full query is complete.



Slot: FULL_SYN
starts with
String: *>benign

*Figure 5.14 Finished query - FULL_SYN starts with *>benign*

7. Leave the search scope set to **Match All** (Boolean AND).
8. Click the **Search** button Search.
9. In the **Search Results** pane on the right, do one of the following:
   - Scroll to locate **Benign Breast Neoplasm**, or
   - Type part of the name in the **Search** field below the **Search Results** list.
10. Double-click the **Benign Breast Neoplasm** search result.

    Protégé switches to the **NCI Editor** > **Edit** subtab, which now displays FULL_SYNs, definitions, and qualifiers for the selected class.

Building a Combined Property Query: Preferred_Name and FULL_SYN

This example uses multiple queries with both simple properties (Preferred_Name) and complex properties (FULL_SYN).

Scenario: You want to search for all genes and alleles that contain *BCL* in the Preferred_Name or FULL_SYN properties.

Procedure: To build and execute a combined property query, build each part of the query as explained in the following procedures.

- *Query 1: Preferred_Name starts with bcl*gene* on this page
- *Query 2: FULL_SYN starts with *>bcl*gene* on page 63
- *Query 3: Preferred_Name starts with bcl*allele* on page 64
- *Query 4: FULL_SYN starts with *>bcl*allele* on page 64

*Query 1: Preferred_Name starts with bcl*gene*

To build the first of the four queries in our combined property query example, follow these steps:

1. Click the **Lucene Query** tab.

   As shown in *Figure 5.15*, the Query pane on the left shows the default query.

Figure 5.15 Default query

Your first query will search for genes with `BCL` in the **Preferred_Name** property. Since Preferred_Name is already specified, the first thing you need to do is set the search parameter to **starts with**.

2. Select **starts with** from the parameter list.

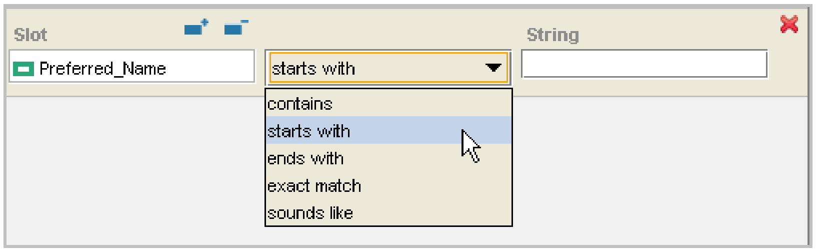
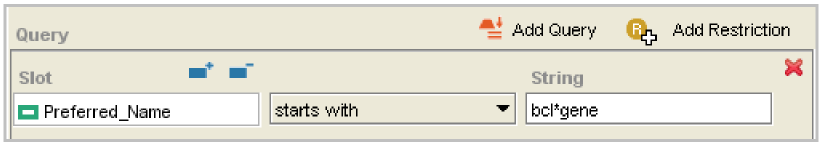


Figure 5.16 Selecting from parameter list

3. In the **String** field, type `bcl*gene`, as shown in *Figure 5.17*.

Tip: Since you are using the **Preferred_Name** property for this query, you can omit the angle brackets.



Preferred_Name | starts with | bcl*gene

*Figure 5.17 Completed query - Preferred_Name starts with bcl*gene*

The first query is complete. Next, you'll add a second query to search for genes with *BCL* in the FULL_SYN property.

*Query 2: FULL_SYN starts with *>bcl*gene*

Prerequisite procedure: *Query 1: Preferred_Name starts with bcl*gene* on page 62

To build the second of the four queries in our combined property query example, follow these steps:

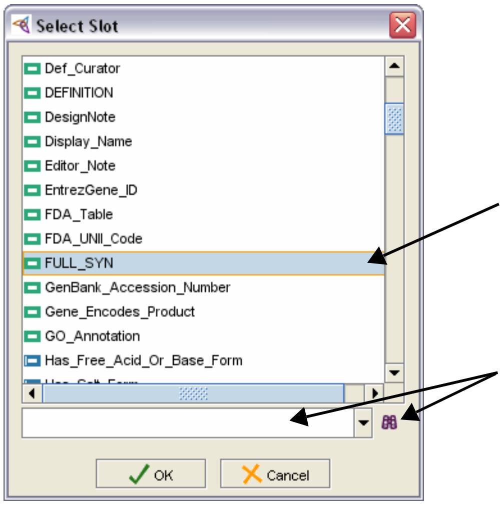
1. Click the **Add Query** button Add Query .

   The new query shows the default query using *Preferred_Name* as the slot. You now need to change the slot to **FULL_SYN**.

2. Click the **Select Slot** button .

   The Select Slot window opens (*Figure 5.18*).

3. Select or search for the **FULL_SYN** property.



Scroll to the item in the list...

...or enter a term and click the **Find** button.

Figure 5.18 Select Slot window

4. Click **OK** to close the window.

5. Change the search parameter to **starts with**.

6. In the **String** field, type `*>bcl*gene`.

   Since FULL_SYN is a complex property, include a preceding angle bracket: `*>bcl*gene`.

*Query 3: Preferred_Name starts with bcl*allele*

Prerequisite procedure: *Query 2: FULL_SYN starts with *>bcl*gene* on page 63

To build the third of the four queries in our combined property query example, follow these steps:

1. Click the **Add Query** Add Query button.
2. Leave **Preferred_Name** as the slot.
3. Change the search parameter to **starts with**.
4. In the **String** field, type `bcl*allele`.

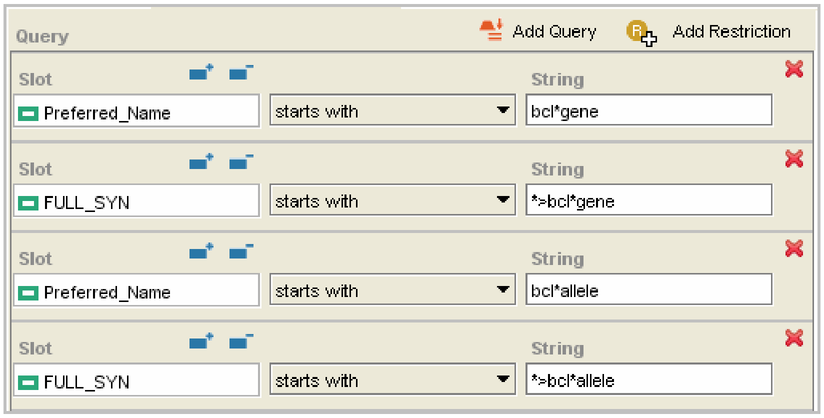
*Query 4: FULL_SYN starts with *>bcl*allele*

Prerequisite procedure: *Query 3: Preferred_Name starts with bcl*allele* on page 64

To build the fourth and final query in our combined property query example, follow these steps:

1. Click the **Add Query** Add Query button.
2. Change the slot to **FULL_SYN**.
3. Change the search parameter to **starts with**.
4. In the **String** field, type `*>bcl*allele`.

   The combined queries should now resemble *Figure 5.19*.



Preferred_Name | starts with | bcl*gene
FULL_SYN | starts with | *>bcl*gene
Preferred_Name | starts with | bcl*allele
FULL_SYN | starts with | *>bcl*allele

Figure 5.19 Combined queries

5. Change the search scope to **Match Any** (Boolean OR).
6. Click the **Search** button Search.
7. Double-click a result to view its details.

Building a Restriction-Based Query

This topic explains how to build an advanced query that has specific restrictions.

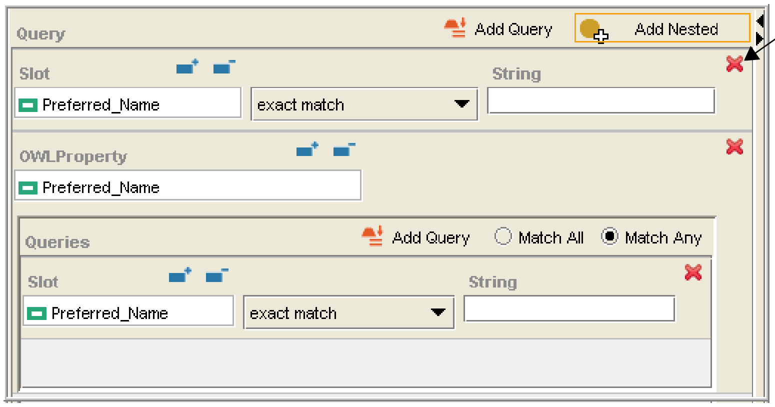
Scenario: You want to find all genes that play a role in *Oncogenesis*.

Procedure: To build and execute a restriction-based query, follow these steps:

1. Click the **Lucene Query** tab.
2. Click the **Add Nested** button Add Nested.

   Protégé places the restriction below the default query. Since you do not need the default query, you can remove it.

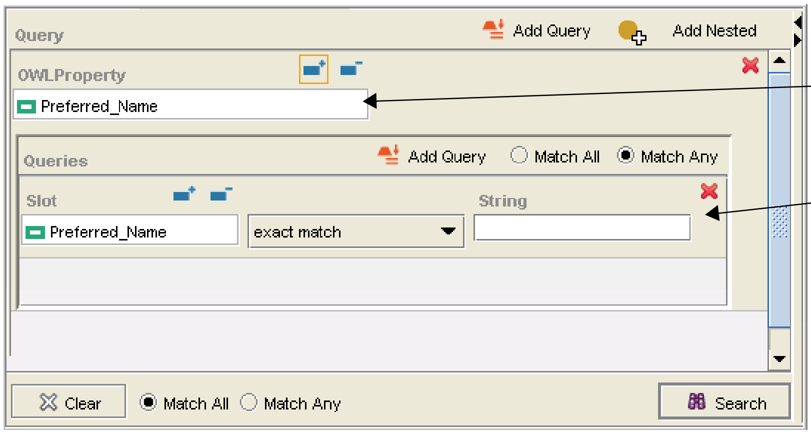
3. Delete the default query by clicking the first instance of the **Remove Query** button ✖ (*Figure 5.20*).



Remove query button

Figure 5.20 Removing the default query

The left side of the Lucene Query tab should now resemble *Figure 5.21*.



Property field

Nested query

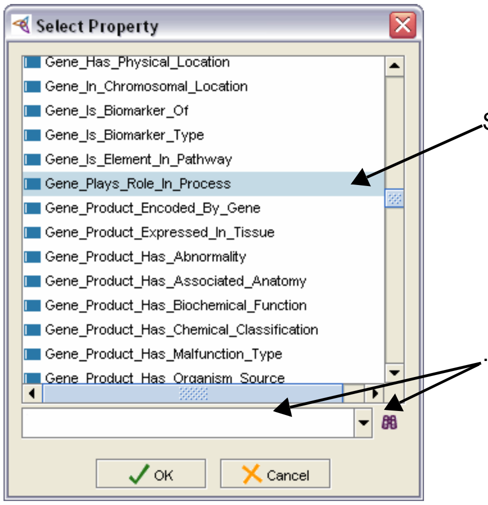
Figure 5.21 Initial setup for restriction-based query

In the next step, you will change the first **Slot** field so that it shows the property *Gene_Plays_Role_In_Process*.

4. Click the **Select Slot** button in the **upper** part of the Query panel.

The Select Property window opens (*Figure 5.22*).

5. Select or search for the following property: *Gene_Plays_Role_In_Process*.



Gene_Has_Physical_Location
Gene_In_Chromosomal_Location
Gene_Is_Biomarker_Of
Gene_Is_Biomarker_Type
Gene_Is_Element_In_Pathway
Gene_Plays_Role_In_Process
Gene_Product_Encoded_By_Gene
Gene_Product_Expressed_In_Tissue
Gene_Product_Has_Abnormality
Gene_Product_Has_Associated_Anatomy
Gene_Product_Has_Biochemical_Function
Gene_Product_Has_Chemical_Classification
Gene_Product_Has_Malfunction_Type
Gene_Product_Has_Organism_Source

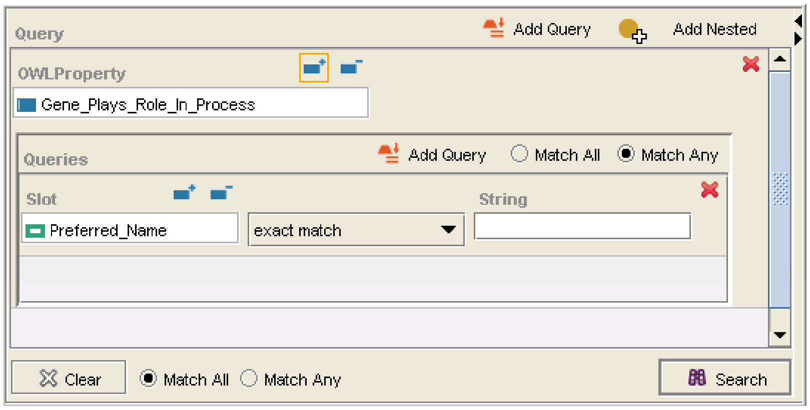
Scroll to the item in the list...

...or enter a term and click the **Find** button.

Figure 5.22 Select property window

6. Click **OK** to close the Select Property window.

   The upper **Slot** field now shows *Gene_Plays_Role_In_Process* (*Figure 5.23*).



OWLProperty: Gene_Plays_Role_In_Process

○ Match All ● Match Any

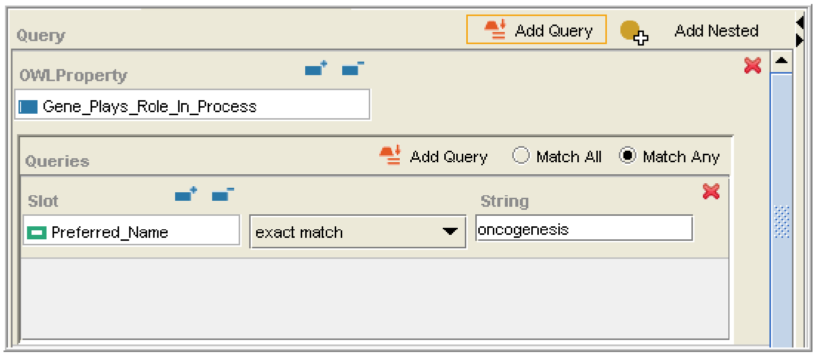
Slot: Preferred_Name | exact match | String:

● Match All ○ Match Any

Figure 5.23 Restriction-based query with search property

7. In the nested Queries panel, leave the slot set to **Preferred_Name**.
8. Leave the parameter set to **exact match**.

9. In the **String** field, type `oncogenesis` (lowercase or uppercase). The query should now resemble *Figure 5.24*.



OWLProperty: Gene_Plays_Role_In_Process

○ Match All ● Match Any

Slot: Preferred_Name

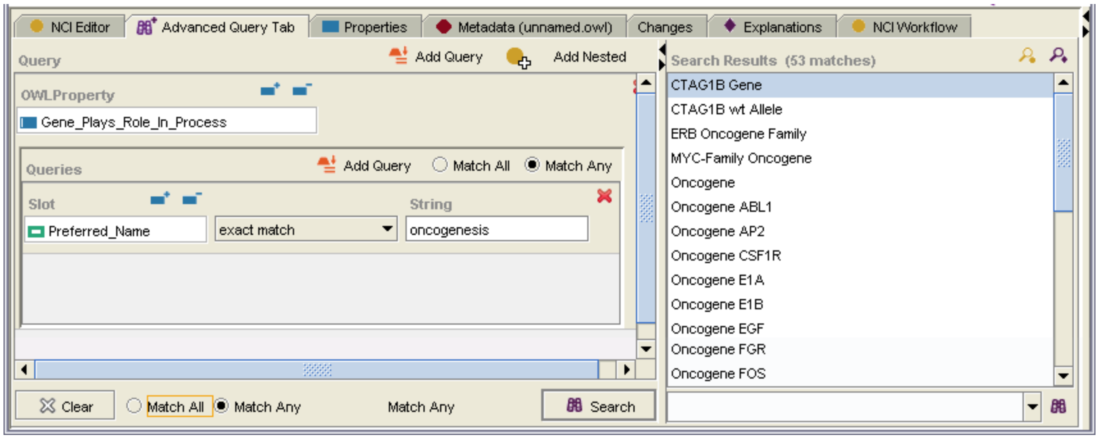
exact match

String: oncogenesis

Figure 5.24 Restriction-based query with completed fields

10. Click the **Search** button Search.

The search results appear on the right panel (*Figure 5.25*).



OWLProperty: Gene_Plays_Role_In_Process

○ Match All ● Match Any

Slot: Preferred_Name

exact match

String: oncogenesis

○ Match All ● Match Any

Match Any

Search Results (53 matches)

- CTAG1B Gene
- CTAG1B wt Allele
- ERB Oncogene Family
- MYC-Family Oncogene
- Oncogene
- Oncogene ABL1
- Oncogene AP2
- Oncogene CSF1R
- Oncogene E1A
- Oncogene E1B
- Oncogene EGF
- Oncogene FGR
- Oncogene FOS

Figure 5.25 Search results for restriction-based query

11. Double-click a result to view its details.

Building an Association-Based Query

This topic explains how to search by association.

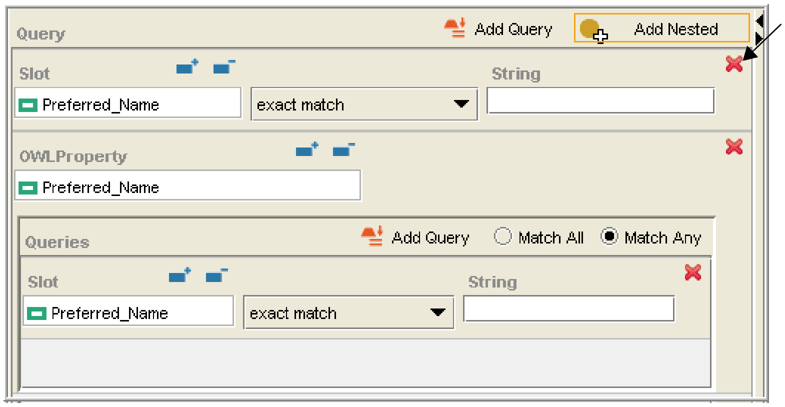
Scenario: You want to find concepts that have an association to another concept that contains a Preferred Name of *FDA established names*.

Procedure: To build and execute an association-based query, follow these steps:

1. Click the **Lucene Query** tab.
2. Click the **Add Nested** button Add Nested.

   The association appears below the default query. Since you do not need additional search terms for this example, you can remove the default query.

3. Delete the default query by clicking the first instance of the **Remove Query** button ✖ (*Figure 5.26*).



Slot: Preferred_Name
exact match
OWLProperty: Preferred_Name
Slot: Preferred_Name
exact match

Remove query button

Figure 5.26 Removing the default query

   In the next step, you will change the first **Slot** field so that it shows the property *Concept_In_Subset*.

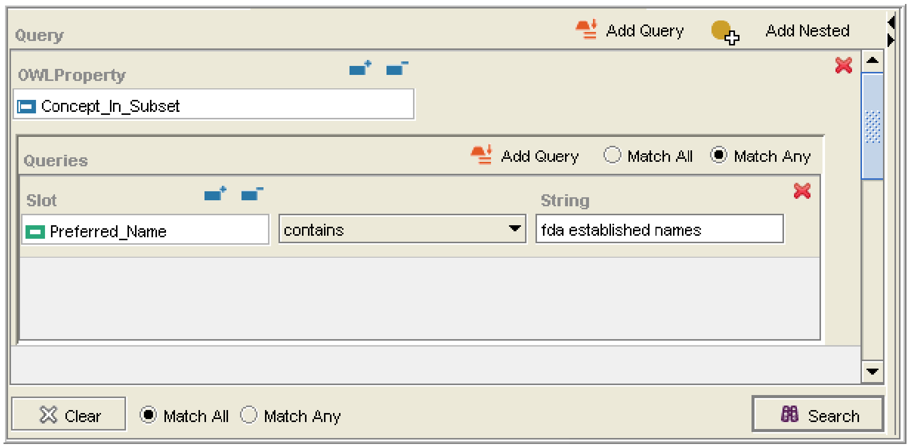
4. Click the **Select Slot** button in the **upper** part of the Query panel.

   The Select Property window opens.

5. Select or search for the following property: *Concept_In_Subset*.
6. Click **OK** to close the Select Property window.
7. In the nested Queries panel, leave the slot set to **Preferred_Name**.
8. Set the search parameter to **contains**.
9. In the **String** field, type `fda established names` (lowercase or uppercase).

   The query should now resemble *Figure 5.27*, with the *upper* **Slot** field showing *Concept_In_Subset*, and the lower fields showing the following:

   - **Slot:** Preferred_Name
   - **Parameter:** contains
   - **String:** `fda established names`



OWLProperty: Concept_In_Subset

Queries: ○ Match All ● Match Any

Slot: Preferred_Name | contains | String: fda established names

● Match All ○ Match Any

Figure 5.27 Association-based query with search property

10. Click the **Search** button Search.

    The search results appear on the right panel.

11. To view details for a result, double-click the result.

Exporting Your Search Results

Using the NCI Protégé Advanced Query plug-in, you can export search results and save them to a text file. You can specify the included properties, the column headings, and the delimiters to be used in the file. You can then open the saved text file in Microsoft Excel, save it as an Excel workbook (.xls), and modify the file as needed.

After building and executing a query, complete the following procedures in the following order:

- *Starting the Export* on page 70
- *Saving Your Search Results as an Excel Workbook* on page 72

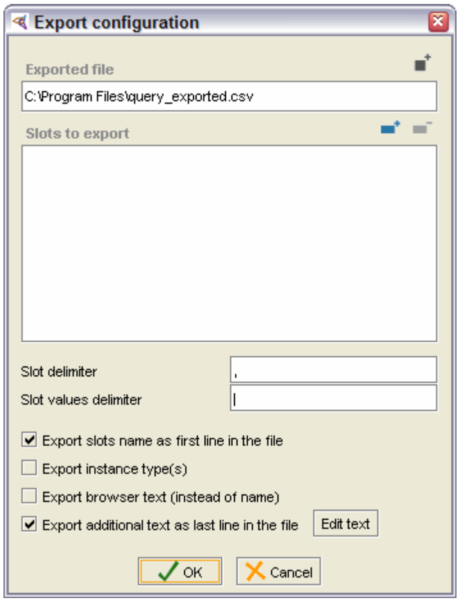
Note: The query used in the examples is *Preferred_Name contains signal*.

Starting the Export

After examining the search results to ensure that they meet your requirements, follow these steps to export the results:

1. Click the **Export Slot Values to File** button (bold capital letter E) E just above the results list, on the far right.

   The Export configuration window opens. A default file path and name appears in the **Exported file** field (*Figure 5.28*).

   Exported file: C:\Program Files\query_exported.csv

   Slot delimiter: ,

   Slot values delimiter: |

   ☑ Export slots name as first line in the file

   ☐ Export instance type(s)

   ☐ Export browser text (instead of name)

   ☑ Export additional text as last line in the file

   Figure 5.28 Export configuration window

2. (Optional) If you prefer to rename the file and save it to a different directory, follow these steps:

   a. Click the **Browse for File** button.

   b. Navigate to the desired directory.

   c. Change the **Files of Type** list to show **All Files**.

   d. (Optional) Change the file name to your preferred name.

   e. Click the **Select** button.

3. To export only specified properties, click the **Add slots** button, located above and to the right of the **Slots to export** field.

   The Select slots to export (multiple selection) window opens. *Multiple selection* means that you can select multiple, non-consecutive properties by holding down the CTRL key and clicking the properties that you want.

   Note: If you select all properties by CTRL-clicking and then clicking **OK**, the properties appear in alphabetical order in both the **Slots to export** list and in the output file. If you want to control the order, select each slot one at a time, click **OK** to close the window, and verify that the selected slot appears in the **Slots to export** field. If you would rather not select the individual slots, you can always re-arrange the column order in the Excel output file.

4. Leave the **Slot delimiter** set to a comma.

   The slot delimiter separates the various properties in the output file.

5. Leave the **Slot values delimiter** field set to a **Pipe** symbol (|).

   The slot values delimiter separates property values in the output file.

6. Check the box labeled **Export slots name as first line in the file**.

   This setting ensures that column headings appear in the output file.

7. (Optional) Check the box labeled **Export instances type(s)**.

   This setting ensures that the exported file includes a column showing the type of exported instance.

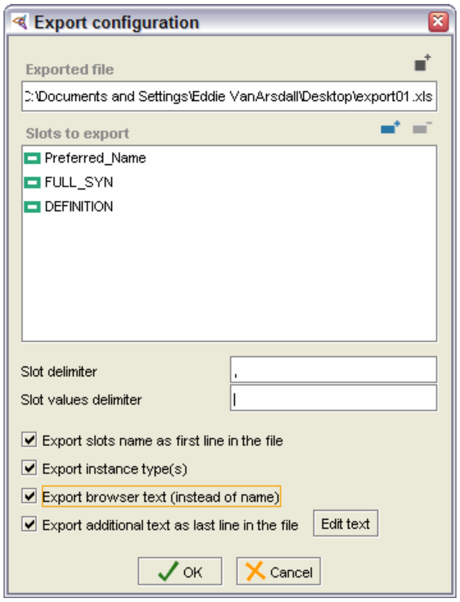
8. (Optional) Check the box labeled **Export browser text (instead of name)**.

   This setting enhances the readability of the output file by using the more user-friendly slot names, such as Preferred_Name, FULL_SYN, and DEFINITION.

9. (Optional) Check the box labeled **Export additional text as last line in file**.

   This step appends the query syntax to the end of the file.
   Example: `(Preferred_Name contains *signal*)`

Figure 5.29 shows the completed Export configuration window.



Exported file

:\Documents and Settings\Eddie VanArsdall\Desktop\export01.xls

Slots to export

Preferred_Name
FULL_SYN
DEFINITION

Slot delimiter ,
Slot values delimiter |

☑ Export slots name as first line in the file
☑ Export instance type(s)
☑ Export browser text (instead of name)
☑ Export additional text as last line in the file

Figure 5.29 Export configuration window with selected slots and settings

10. Click **OK**.

    If the export runs successfully, an Export successful message window opens.

11. Click the **Close** button to close the message window.

Saving Your Search Results as an Excel Workbook

Prerequisite procedure: *Starting the Export* on page 70

To save an exported query file as an Excel workbook, follow these steps:

1. Open Microsoft Excel.
2. Open the saved .csv file.
3. When the Text Import Wizard launches, use the following Wizard settings:
    a. Step 1 of 3 screen: Select **Delimited** as the file type, then click **Next**.
    b. Step 2 of 3 screen under **Delimiters**:
        — Clear the **Tab** check box.
        — Select the **Other** check box, then type a pipe symbol (|) in the box.
        — Click **Next**.
    c. Step 3 of 3 screen: Click **Finish**.

        The delimited data now appears in workbook columns.
4. Save the file in Microsoft Excel (.xls) format.

    You can now use various Excel features to modify the output.

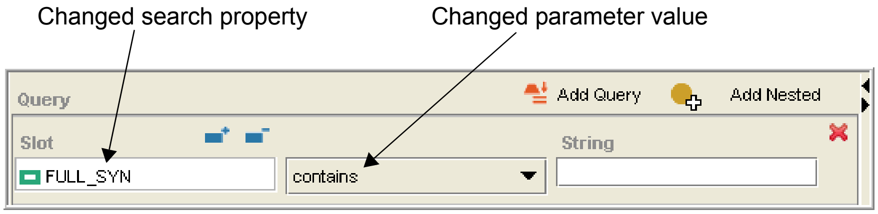
Configuring the Lucene Query tab

The Lucene Query tab uses *Preferred_Name* as the default search property (or *slot*) and *exact match* as the default parameter. You can reconfigure the search defaults using the Protégé Configure window. The results are stored in the *protege.properties* file at `C:\Program Files\Protege.Client-1.2.x`.

The following topics explain how to change both the default search property and the default search parameter. The example shows you how to make the following changes:

- Change the default search property from *Preferred_Name* to *FULL_SYN*.
- Change the default search parameter from *exact match* to *contains*.

Once you make these changes, the default query on the Lucene Query tab should resemble *Figure 5.30*.



Slot: FULL_SYN

contains

String:

Figure 5.30 Default query with new search property and search parameter

To configure the Lucene Query tab, complete the following procedures in the following order:

- *Changing the Default Search Property* on this page
- *Changing the Default Search Parameter* on page 75
- *Verifying Search Configuration Changes* on page 76

Changing the Default Search Property

Prerequisite topic: *Configuring the Lucene Query tab*

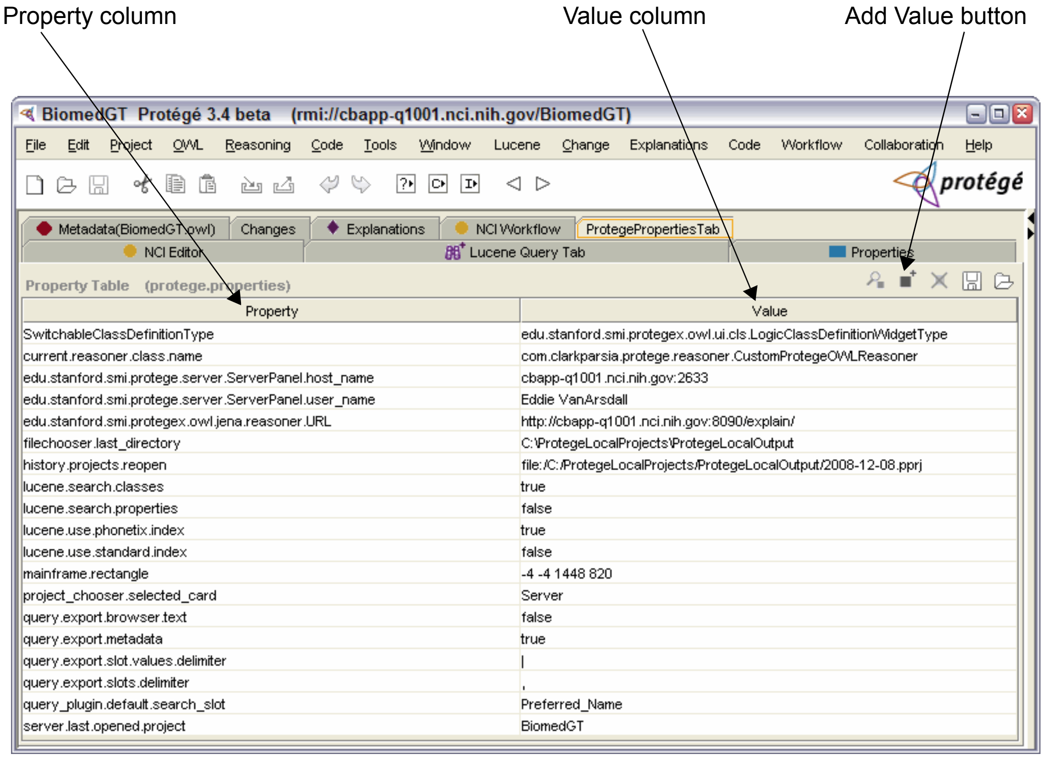
To change the default query, you first need to change the default search property from *Preferred_Name* to *FULL_SYN*.

To change the default search property, follow these steps:

1. Select the following menu command: **Project** > **Configure**.
2. In the Configure window, select the **Tab Widgets** tab.
3. Check the box to the left of the **ProtegePropertiesTab** widget.
4. Click **OK**.

   The ProtegePropertiesTab widget now appears on the far right (*Figure 5.31* on page 74). Note the **Property** and **Value** columns. The combined contents of these two columns are known as *name-value* pairs. The **Property** column shows the property name, and the **Value** column shows the corresponding value for the property.

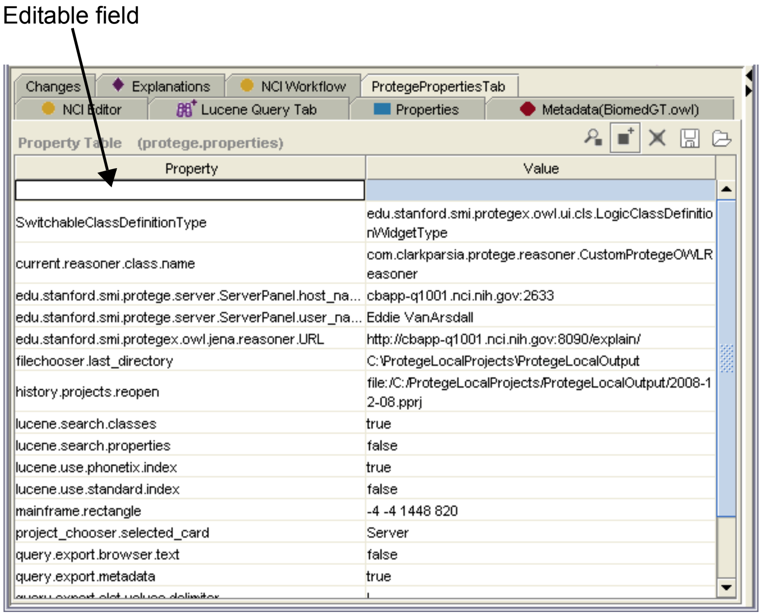
5. Click the **Add Value** button ▪⁺ (*Figure 5.31*).



Property	Value
SwitchableClassDefinitionType	edu.stanford.smi.protegex.owl.ui.cls.LogicClassDefinitionWidgetType
current.reasoner.class.name	com.clarkparsia.protege.reasoner.CustomProtegeOWLReasoner
edu.stanford.smi.protege.server.ServerPanel.host_name	cbapp-q1001.nci.nih.gov:2633
edu.stanford.smi.protege.server.ServerPanel.user_name	Eddie VanArsdall
edu.stanford.smi.protegex.owl.jena.reasoner.URL	http://cbapp-q1001.nci.nih.gov:8090/explain/
filechooser.last_directory	C:\ProtegeLocalProjects\ProtegeLocalOutput
history.projects.reopen	file:/C:/ProtegeLocalProjects/ProtegeLocalOutput/2008-12-08.pprj
lucene.search.classes	true
lucene.search.properties	false
lucene.use.phonetix.index	true
lucene.use.standard.index	false
mainframe.rectangle	-4 -4 1448 820
project_chooser.selected_card	Server
query.export.browser.text	false
query.export.metadata	true
query.export.slot.values.delimiter	|
query.export.slots.delimiter	,
query_plugin.default.search_slot	Preferred_Name
server.last.opened.project	BiomedGT

Figure 5.31 ProtegePropertiesTab widget

An editable text field appears in the **Property** column (*Figure 5.32*).



Property	Value
SwitchableClassDefinitionType	edu.stanford.smi.protegex.owl.ui.cls.LogicClassDefinitionWidgetType
current.reasoner.class.name	com.clarkparsia.protege.reasoner.CustomProtegeOWLReasoner
edu.stanford.smi.protege.server.ServerPanel.host_na...	cbapp-q1001.nci.nih.gov:2633
edu.stanford.smi.protege.server.ServerPanel.user_na...	Eddie VanArsdall
edu.stanford.smi.protegex.owl.jena.reasoner.URL	http://cbapp-q1001.nci.nih.gov:8090/explain/
filechooser.last_directory	C:\ProtegeLocalProjects\ProtegeLocalOutput
history.projects.reopen	file:/C:/ProtegeLocalProjects/ProtegeLocalOutput/2008-12-08.pprj
lucene.search.classes	true
lucene.search.properties	false
lucene.use.phonetix.index	true
lucene.use.standard.index	false
mainframe.rectangle	-4 -4 1448 820
project_chooser.selected_card	Server
query.export.browser.text	false
query.export.metadata	true
query.export.slot.values.delimiter	|

Figure 5.32 ProtegePropertiesTab widget with editable field

6. In the **Property** field, type the following exactly as shown: `query_plugin.default.search_slot`

   This represents the property name in the name-value pair.

7. Press TAB to move to the empty field in the **Value** column on the right.

   This step activates the **Value** field for editing and simultaneously reorders the Property-Value list.

8. In the **Value** field, type `FULL_SYN`.
9. Press TAB to move the cursor out of the field.
10. Leave the ProtegePropertiesTab widget displayed for the next procedure.
11. Continue to *Changing the Default Search Parameter* on page 75.

Changing the Default Search Parameter

Prerequisite procedure: *Changing the Default Search Property* on page 73

To change the name-value pair for the default search parameter, follow these steps:

1. With the ProtegePropertiesTab still open, click the **Add Value** button.

   A new, editable field appears in the **Property** column on the left.

2. In the **Property** field, type the following exactly as shown: `query_plugin.default.search_type.String`
3. Press TAB to enable the empty field in the **Value** column on the right.

   This step activates the **Value** field for editing and simultaneously reorders the Property-Value list.

4. In the **Value** field, type `contains`.
5. Press TAB to move the cursor out of the field.
6. Click **OK** to close the Configure window.
7. To close the With the ProtegePropertiesTab widget, follow these steps:
   a. Select the following menu command: **Project** > **Configure**
   b. Select the **Tab Widgets** tab.
   c. Clear the check box to the left of the **ProtegePropertiesTab** widget.
   d. Click **OK**.

Note: Upon exiting, Protégé will write these new values to a properties file on your hard drive. For more information, see *Verifying Search Configuration Changes* on page 76.

Verifying Search Configuration Changes

Prerequisite procedures:

- *Changing the Default Search Property* on page 73
- *Changing the Default Search Parameter* on page 75

The best way to verify that the new configuration is in effect is to check the default query on the Lucene Query tab. However, changing properties and values in the Configure file window does not automatically refresh the Protégé interface; thus, you might still see the original query using *Preferred Name* with exact matching.

To force Protégé to refresh its interface, follow these steps:

1. Click the **Edit** subtab.
2. To refresh the interface, select a class in the Class Browser on the left, ensuring that you select a different class from the one that is currently displayed.
3. Click the **Lucene Query tab**. The new default search property and parameter should now show on the left side of the tab.

Note: If you still do not see a difference, try closing and restarting Protégé.

Understanding Where Search Properties and Values Are Stored

Configuration information is stored in the `protege.properties` file at `C:\Program Files\Protege.Client-1.2.x`. Within the file, the following entries represent the default query configuration:

```
query_plugin.default.search_slot=Preferred_Name

query_plugin.default.search_type.String=exact match
```

Note: The property name appears on the left side of the equal sign, and the property value appears on the right side. The name and value correspond to the **Property** and **Value** columns in the Configure file window that is used to configure the default search property and parameter.

Note that the first string refers to the *search_slot*, meaning the default search property. The second string refers to the *search_type*, which is followed by a period and the word *String*. This denotes the data type being used in the search.

If you configured your settings as specified in *Configuring the Lucene Query tab* on page 73, the following entries were saved to the *protege.properties* file:

```
query_plugin.default.search_slot=FULL_SYN

query_plugin.default.search_type.String=contains
```

Available Data Types from SMI

Stanford Medical Informatics (SMI) has developed the **Advanced Query** plug-in so that it enables you to configure Protégé to use several data types. Currently, the NCI is using only the *String* data type, so the examples given in the previous topics use strings. Other types may be used at the NCI in the future.

Table 5.2 lists the possible data types with the property name and accepted values for each. In the **Possible Values** column, *is* means *equals*.

Data Type	***Property Name***	***Possible Values***
Any	query_plugin.default.search_type.Any	contains starts with ends with exact match sounds like
Boolean	query_plugin.default.search_type.Boolean	is
Class	query_plugin.default.search_type.Cls	contains
Float	query_plugin.default.search_type.Float	is greater than less than
Instance	query_plugin.default.search_type.Instance	contains
Integer	query_plugin.default.search_type.Integer	is greater than less than
String	query_plugin.default.search_type.String	contains starts with ends with exact match sounds like
Symbol	query_plugin.default.search_type.Symbol	is

Table 5.2 Search data types implemented by SMI

CHAPTER

6

WORKING WITH CLASSES: THE BASICS

This chapter provides an overview of classes in NCI Protégé and explains how to create and tree a class.

Topics in this Chapter

- *About Classes* on this page
- *Examining Class Information* on page 82
- *Reviewing Changes As You Work* on page 87
- *Creating a Class* on page 88
- *Treeing a Class* on page 91
- *Viewing Classes Connected by part_of Relations* on page 96

About Classes

The basic unit of information in the NCI Thesaurus is a *concept*. A concept describes sets of individuals in a given domain.

Protégé refers to concepts as *classes*. Like a concept, a class has a name, belongs to a namespace, and exists in relation to other classes.

Class Identifiers

In the NCI Thesaurus, class *names* and class *codes* are the principal identifiers for classes. Each class in the Thesaurus must therefore have a unique name. Once you create a class, you cannot change its name, so make sure that you name the class as intended and that you spell its name correctly.

OWL-compliant class names can have underscores and dashes but no other punctuation. For example, *Antigen_Gene*, *Cell-Cell_Adhesion*, and *Radiation-Induced_Intracranial_Meningioma* are all OWL-compliant names.

Each class also has a *preferred name*. This is the name that users see, and it can include spaces. Though you cannot change a class name, you *can* change the preferred name. Both class names and preferred names are generally singular.

Class Relationships

Protégé uses a tree analogy to describe class relationships. The Class Hierarchy available in the Class Browser has a root class called *owl:Thing.* All other classes have parent-child relationships. In the Protégé vernacular, parent classes are *superclasses*, and child classes are *subclasses*.

Class Properties

Although *Working with Properties* (page 99) explains how to add and edit properties, this chapter introduces you to properties because they are integral to working with classes. You also need to understand the subtabs that represent properties when you create and tree classes.

Essentially, properties describe a class:

- *Simple properties* such as a preferred name provide text string values.
- *Complex properties* provide both a value and additional information about the value, expressed using *property qualifiers*.

The following topics discuss two types of complex properties.

About the FULL_SYN Property

A *FULL_SYN*, or full synonym, can include the qualifiers listed in *Table 6.1*.

Qualifier	***Description***
Term Name	Text strings representing name variations.
Term Group	Abbreviations for term types, the most common of which are *Preferred Term* (PT) and *Synonym* (SY). When you edit a property, you can select an abbreviation from the term group list.
Term Source	NCI groups or outside contributors who have supplied terms to the EVS and who need to preserve those terms for their own purposes. Terms from sources other than the NCI should not be changed without permission, nor should they remain in a retired concept.
Source Code	Only applies to specific sources; not required by the NCI.
Lang	Optional; not currently used.

Table 6.1 Qualifiers and their descriptions

About the Definition Property

The *Definition* property is the official NCI definition for a class. It is accompanied by three qualifiers:

- *Definition_Review_Date*
- *def-source*
- *Definition_Reviewer_Name*

As an NCI editor, you need to ensure that each class has one good technical NCI definition (DEFINITION), written using specified guidelines (usually posted online). Some classes may also have alternate definitions (ALT_DEFINITION) from other sources such as NCI-GLOSS.

Note: You may sometimes need to add descriptive information for a class when the information isn't actually a definition. For example, you may need to add instructions to a coder regarding how and when a class will be applied, notes to another editor regarding why the class is treed in a specific location, or notes about any additional information that is needed. In this case, you will use an Editor's Note or Design Note. These are used in *Using the PreMerge Subtab to Flag Classes for a Merge* on page 126 and *Using the PreRetire Subtab to Flag a Class for Retirement* on page 135.

Guidelines for Writing Definitions

When writing definitions, observe the rules in this topic.

Quoted Definitions

- If you quote word for word from a specific source, cite the source (for example, *American Heritage Dictionary*, *On-line Medical Dictionary*).
- If you use a definition from a specific source but slightly change it, cite it as *from {source name}*.
- If you use a source as a recognizable base and elaborate further, cite the source as *from {source name} and NCI*.

Rewritten and Multiple-Source Definitions

- If you rewrite a definition from a source so that it is hardly recognizable, attribute the definition as an NCI source.
- If you write a definition from various sources and the definition no longer resembles any of those sources, attribute the definition as an NCI source.
- If you write an original definition, attribute it as an NCI source.

Definitions Using Journal Articles as Sources

Cite the source as follows:

Jaju et al. Genes Chromosomes Cancer 1998. 22:251-256

Definitions Using Websites as Sources

Provide enough information so that the reader can find and evaluate the source.

Examples: *Lymphoma Information Network Glossary*, *MedicineNet*, *Chemical and Physical Carcinogenesis Branch*, DCB home page.

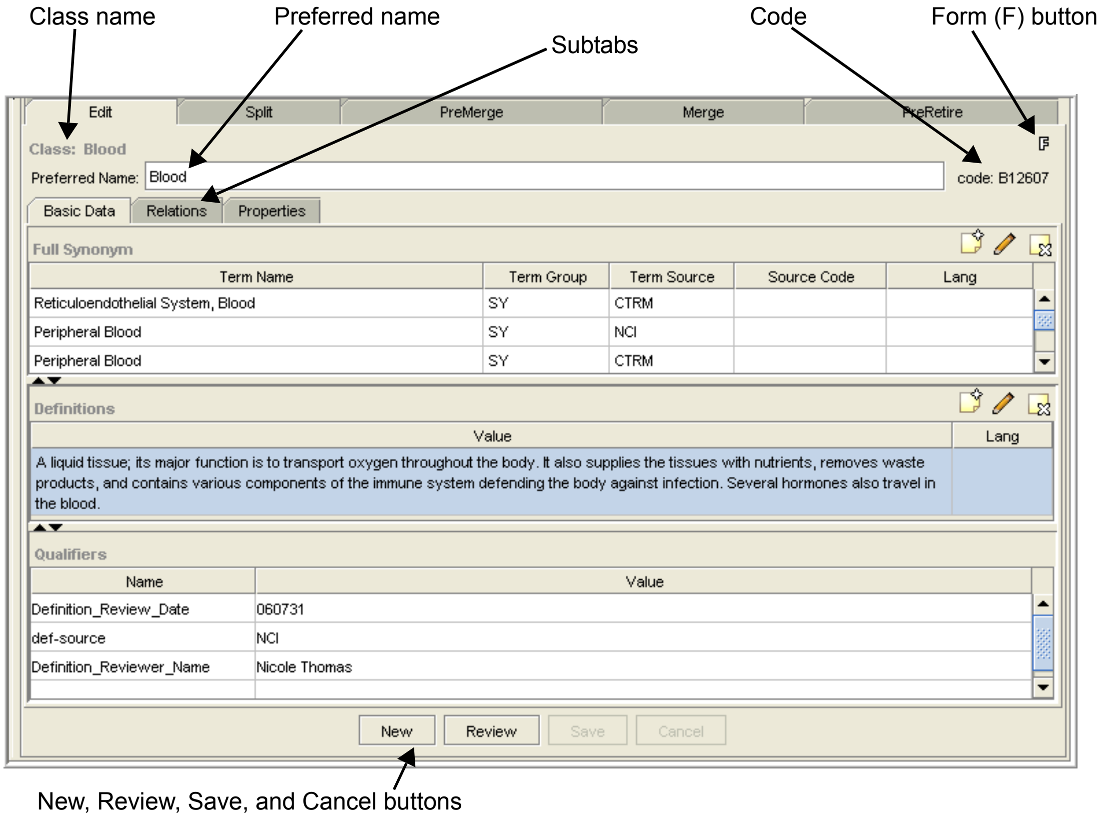
You can also provide a URL, such as http://www.lef.org/prod_hp/abstracts/biostimabs.html.

Definitions from Other Sources

- *Stedman* definitions are for internal use only. If you see an NCI definition citing STED, rewrite it or replace it as necessary.
- *Dorland* and *Devita* are copyright protected and should not be used. If you find definitions from these sources, rewrite or replace them.

Examining Class Information

When you select a class in the Class Browser or from a search results list, NCI Protégé displays the NCI Editor tab > Edit subtab. This subtab shows annotation and relation data for the selected class using three additional subtabs of its own: Basic Data, Relations, and Properties (*Figure 6.1*).



Class: Blood

Preferred Name: Blood

code: B12607

Full Synonym

Term Name	Term Group	Term Source	Source Code	Lang
Reticuloendothelial System, Blood	SY	CTRM		
Peripheral Blood	SY	NCI		
Peripheral Blood	SY	CTRM		

Definitions

Value	Lang
A liquid tissue; its major function is to transport oxygen throughout the body. It also supplies the tissues with nutrients, removes waste products, and contains various components of the immune system defending the body against infection. Several hormones also travel in the blood.	

Qualifiers

Name	Value
Definition_Review_Date	060731
def-source	NCI
Definition_Reviewer_Name	Nicole Thomas

Figure 6.1 Edit subtab

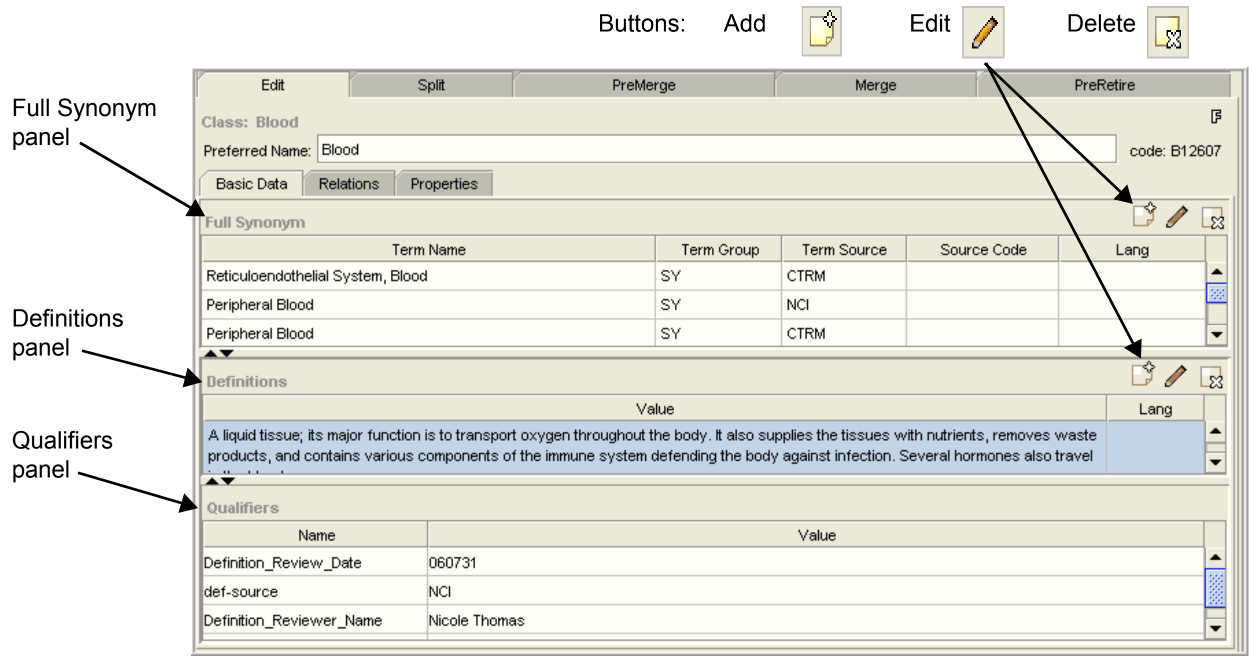
Table 6.1 describes the main features of the Edit subtab.

Feature	**Description**
Class Name, Preferred Name, and Code	Collectively known as the header area.
Form button	Used by server administrators to set up Protégé projects.
Basic Data, Relations, and Properties subtabs	Show detailed information about the class. For more information, see the following topics: • *Basic Data Subtab* on page 83 • *Relations Subtab* on page 84 • *Properties Subtab* on page 85
New button	Creates a new subclass.
Review button	Opens a window that shows changes made to a class, even before you save them. For more information, see *Reviewing Changes As You Work* on page 87.
Save button	Saves your changes.
Cancel button	Discards unsaved changes.

Table 6.2 Edit subtab features

Basic Data Subtab

The Basic Data subtab shows full synonym annotation properties, definition annotation properties, and definition qualifiers. Each property type has its own panel (*Figure 6.2*).



Class: Blood

Preferred Name: Blood

code: B12607

Full Synonym

Term Name	Term Group	Term Source	Source Code	Lang
Reticuloendothelial System, Blood	SY	CTRM		
Peripheral Blood	SY	NCI		
Peripheral Blood	SY	CTRM		

Definitions

Value	Lang
A liquid tissue; its major function is to transport oxygen throughout the body. It also supplies the tissues with nutrients, removes waste products, and contains various components of the immune system defending the body against infection. Several hormones also travel	

Qualifiers

Name	Value
Definition_Review_Date	060731
def-source	NCI
Definition_Reviewer_Name	Nicole Thomas

Figure 6.2 Basic Data subtab

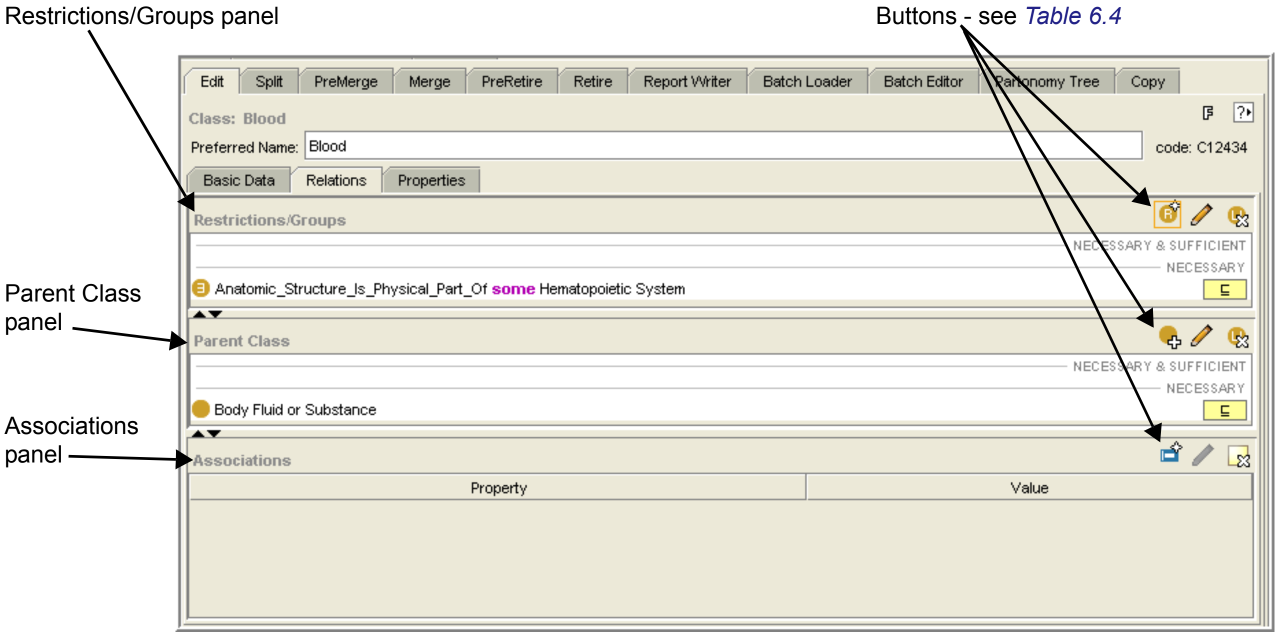
Table 6.3 describes the main features of the Basic Data subtab.

Feature	**Description**
Add, Edit, and Delete buttons	Used to add, edit, or delete a full synonym or definition.
Full Synonym panel	Shows full synonym values.
Definitions panel	Shows definition values.
Qualifiers panel	Shows qualifier values for definition properties.

Table 6.3 Basic Data subtab features

Relations Subtab

The Relations subtab shows values for restrictions, values for parent classes, and properties and values for associations. The layout includes a panel for each category (*Figure 6.3*).



Class: Blood

Preferred Name: Blood

code: C12434

Anatomic_Structure_Is_Physical_Part_Of **some** Hematopoietic System

Body Fluid or Substance

Figure 6.3 Relations subtab

Table 6.4 describes the main features of the Relations subtab.

Feature	**Description**
Restrictions/Groups panel	Shows restriction values.
Add, Edit, and Delete Restriction/Group buttons	Used to create, edit, or delete a restriction or group.
Parent Class panel	Shows classes that are parents of the selected class.

Table 6.4 Relations subtab features

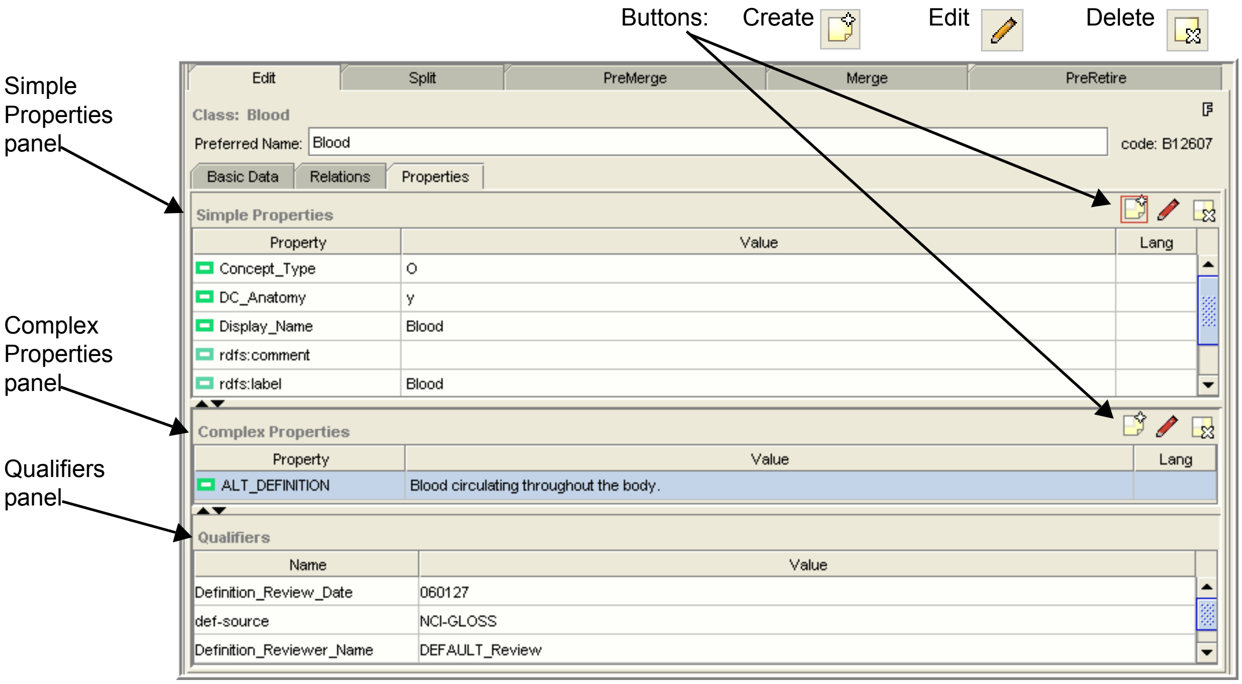
Feature	**Description**
Add, Edit, and Delete Parent Class buttons	Used to create, edit, or delete a superclass.
Associations panel	Displays object-valued properties.
Add, Edit, and Delete Object-Valued Property buttons	Used to create, edit, or delete an object-valued property.

Table 6.4 Relations subtab features

Properties Subtab

The Properties subtab shows simple properties, complex properties, and qualifier values for complex properties (*Figure 6.4*). The layout includes a panel for each property type.

Note: Additional properties are available on the Basic Data subtab(page 83).



Class: Blood

Preferred Name: Blood

code: B12607

Simple Properties

Property	Value	Lang
Concept_Type	O	
DC_Anatomy	y	
Display_Name	Blood	
rdfs:comment		
rdfs:label	Blood	

Complex Properties

Property	Value	Lang
ALT_DEFINITION	Blood circulating throughout the body.	

Qualifiers

Name	Value
Definition_Review_Date	060127
def-source	NCI-GLOSS
Definition_Reviewer_Name	DEFAULT_Review

Figure 6.4 Properties subtab

Table 6.5 lists the main features of the Properties subtab.

Feature	**Description**
Simple Properties panel	Shows simple property values.
Create, Edit, and Delete buttons	Used to create, edit, or delete simple properties.
Complex Properties panel	Shows complex property values.

Table 6.5 Properties subtab features

Feature	***Description***
Create, Edit, and Delete buttons	Used to create, edit, and delete complex properties.
Qualifiers panel	Shows qualifier values for complex properties.

Table 6.5 Properties subtab features

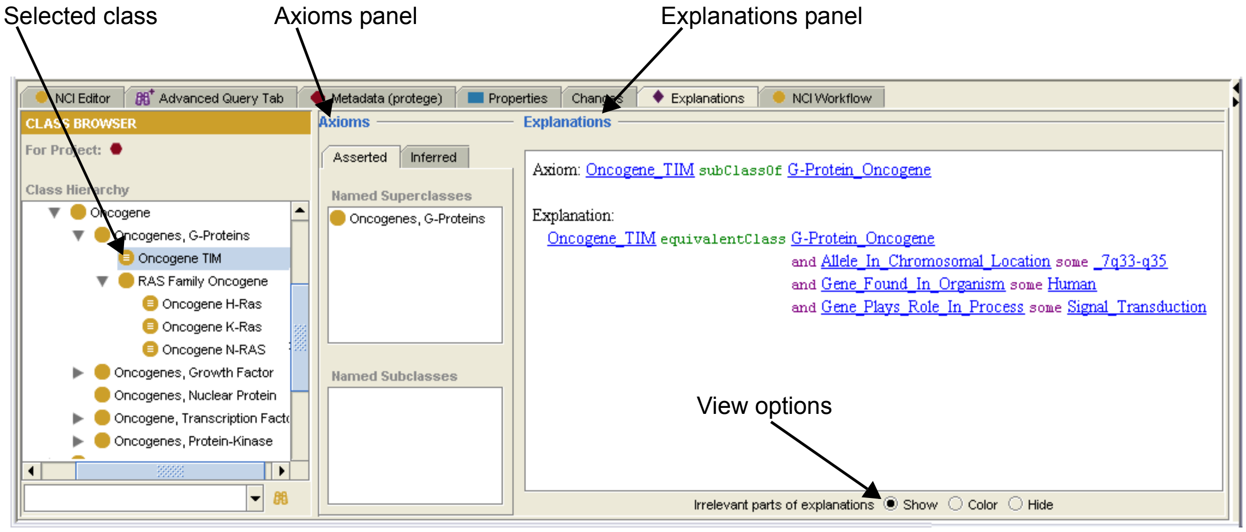
Explanations Tab

If you generate an inconsistent class, use the Explanations tab (*Figure 6.5*) to find out why. This tab explains why a class was not properly classified by the reasoner.

To use the Explanations tab, follow these steps:

1. Using the Class Browser, select the class that you want to examine.
2. Note the following features:
   - Named superclasses appear in the upper frame of the Axioms panel, and named subclasses appear in the lower frame. This panel also offers two tabs: **Asserted** and **Inferred**, which enable you to switch between views.
   - The explanation appears in the Explanations panel on the far right.
3. Use the three option buttons below the Explanations panel to change the view:
   - Click **Show** to see a full explanation with all restrictions.
   - Click **Color** to leaves the restrictions visible but de-emphasize them by showing them in light gray.
   - Click **Hide** to see only the simple explanation and hide the restrictions. This option also leaves the word *and*, followed by an ellipsis (*and...*) as an indicator that there is more to see.

Figure 6.5 shows the features of the Explanations tab.



Axiom: Oncogene_TIM subClassOf G-Protein_Oncogene

Explanation:
Oncogene_TIM equivalentClass G-Protein_Oncogene
and Allele_In_Chromosomal_Location some _7q33-q35
and Gene_Found_In_Organism some Human
and Gene_Plays_Role_In_Process some Signal_Transduction

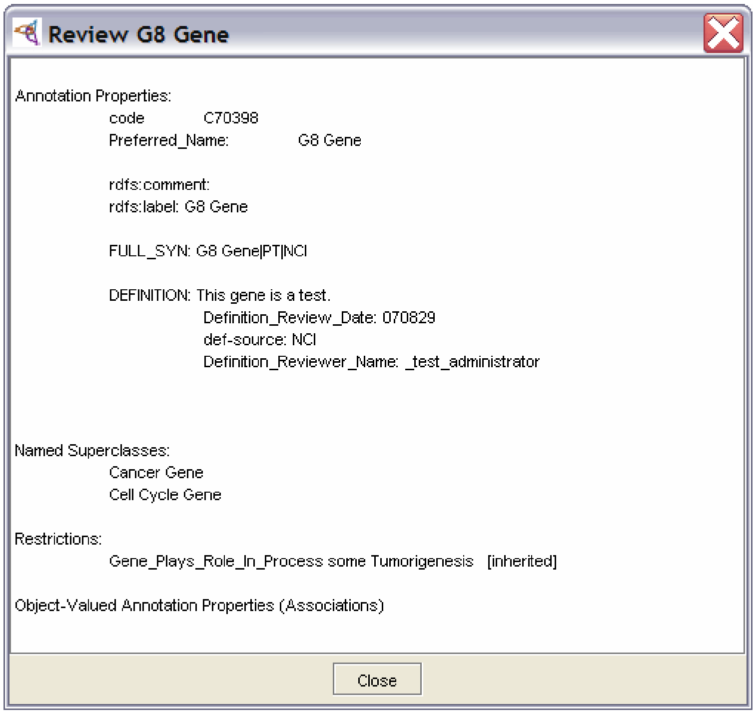
Irrelevant parts of explanations ● Show ○ Color ○ Hide

Figure 6.5 Explanations tab

Reviewing Changes As You Work

As you work with NCI Protégé, you will often use the Basic Data, Relations, and Properties subtabs to view and edit information about classes. You can freely switch between these subtabs without having to save your work each time. Protégé holds your changes in memory as long as you continue working with data shown on these subtabs for the currently selected class.

Before saving changed data, click the **Review** button (the middle button at the bottom of the Edit subtab). This opens a convenient Review window that summarizes the data for the current class, including changes that you have made but have not yet saved.



```
Annotation Properties:
        code        C70398
        Preferred_Name:        G8 Gene

        rdfs:comment:
        rdfs:label: G8 Gene

        FULL_SYN: G8 Gene|PT|NCI

        DEFINITION: This gene is a test.
                Definition_Review_Date: 070829
                def-source: NCI
                Definition_Reviewer_Name: _test_administrator


Named Superclasses:
        Cancer Gene
        Cell Cycle Gene

Restrictions:
        Gene_Plays_Role_In_Process some Tumorigenesis   [inherited]

Object-Valued Annotation Properties (Associations)
```

Figure 6.6 Review window

Tip: Use the **Review** window often to review unsaved changes.

Creating a Class

Before creating a new class, search thoroughly for the class and any equivalent variants. While searching, think of different ways in which the class might be expressed. Search using the class name, preferred name, and FULL_SYN.

Tip: For more information about searching in Protégé, see *Chapter 5, Searching,* on page 51.

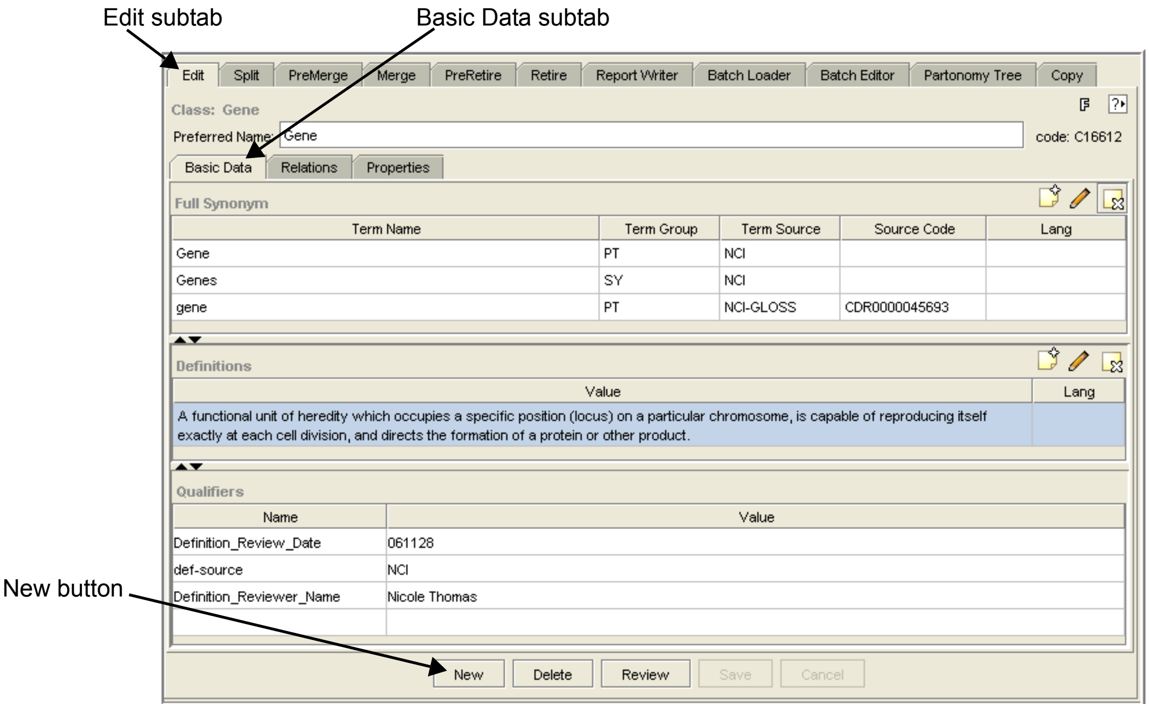
If you find that no existing class is adequate or appropriate, create a class that has *face validity*; that is, ensure that other editors can easily tell how the class differs from existing classes in the database (for example, *Cyclin Kinase Inhibitor* with naturally occurring factors vs. *Cyclin-Dependent Kinase Inhibitor Drug*).

To create a class, follow these steps:

1. Using the Class Browser, select any class except *owl:Thing*.

   Note: The class you select in the hierarchy does not have to be the parent class for the new class. This step is necessary because the button for creating a new class is not enabled unless you select a class other than *owl:Thing*.

2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Basic Data** subtab.
4. Click the **New** button at the bottom of the subtab (*Figure 6.7*).


Edit subtab

Basic Data subtab

New button

Class: Gene

Preferred Name: Gene

code: C16612

Full Synonym

Term Name	Term Group	Term Source	Source Code	Lang
Gene	PT	NCI		
Genes	SY	NCI		
gene	PT	NCI-GLOSS	CDR0000045693	

Definitions

Value	Lang
A functional unit of heredity which occupies a specific position (locus) on a particular chromosome, is capable of reproducing itself exactly at each cell division, and directs the formation of a protein or other product.	

Qualifiers

Name	Value
Definition_Review_Date	061128
def-source	NCI
Definition_Reviewer_Name	Nicole Thomas

Figure 6.7 Creating a new class

The Create Subclass window opens.

5. To select a parent class, click the **Select Superclass** button ● on the right (*Figure 6.8*).

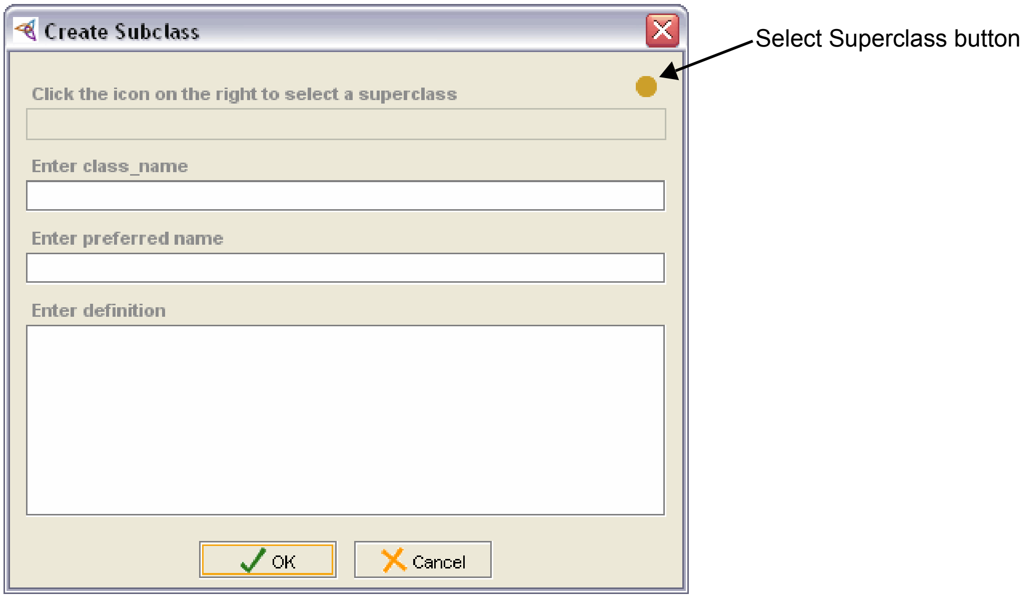


Figure 6.8 Create Subclass window

The Select a superclass window opens (*Figure 6.9*).

6. Select or search for a class.

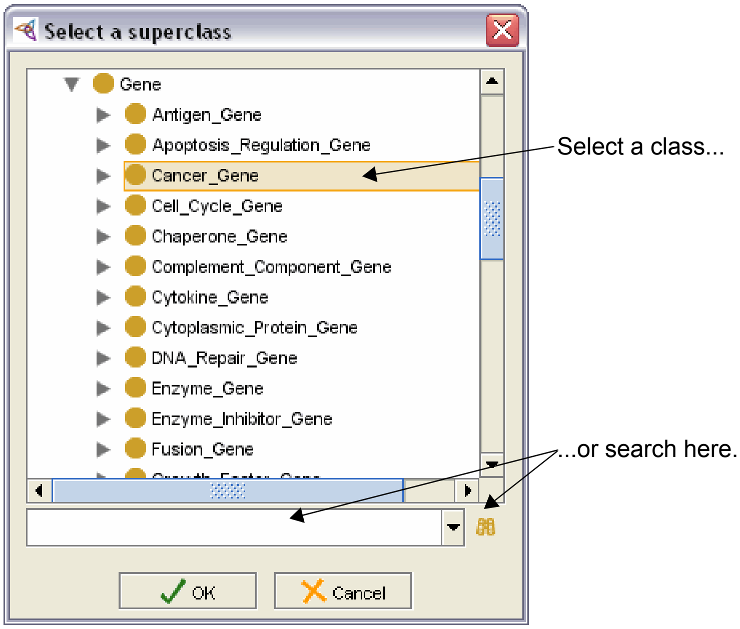


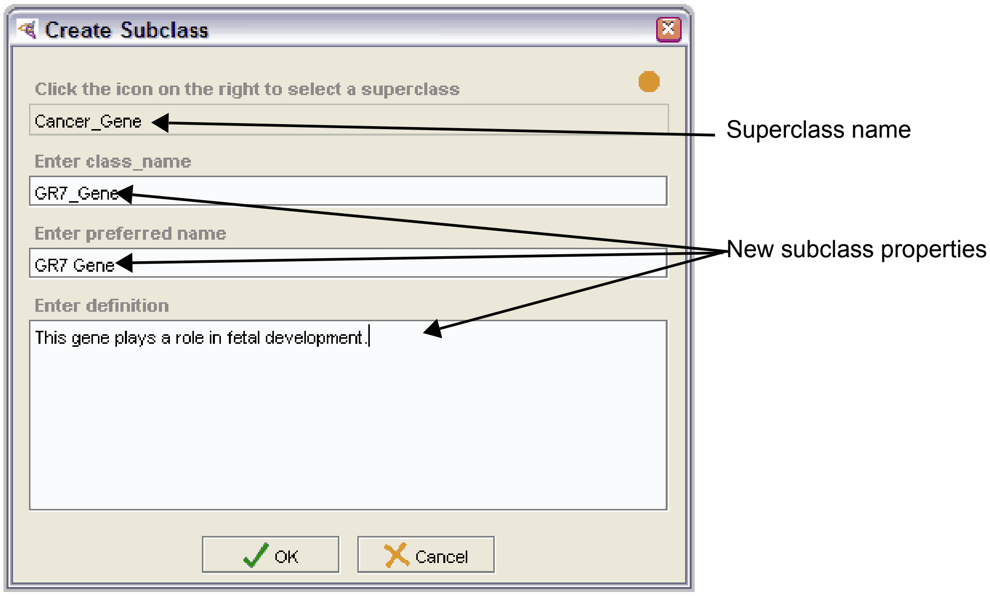
Figure 6.9 Select a superclass window

If you search for a class, the Advanced Query window opens. Select a search result, then click **OK** to close the window and return to the Select a superclass window.

7. Click **OK** to close the Select a superclass window and return to the Create Subclass window.

   The name of the selected class now appears in the upper part of the window (*Figure 6.10*).

8. Enter a class name, preferred name, and definition.
9. (Optional) If you wrote the definition in a word processor, copy and paste it into the **Enter definition** field.
10. Verify that the Create Subclass window now resembles *Figure 6.10*.



Click the icon on the right to select a superclass
Cancer_Gene

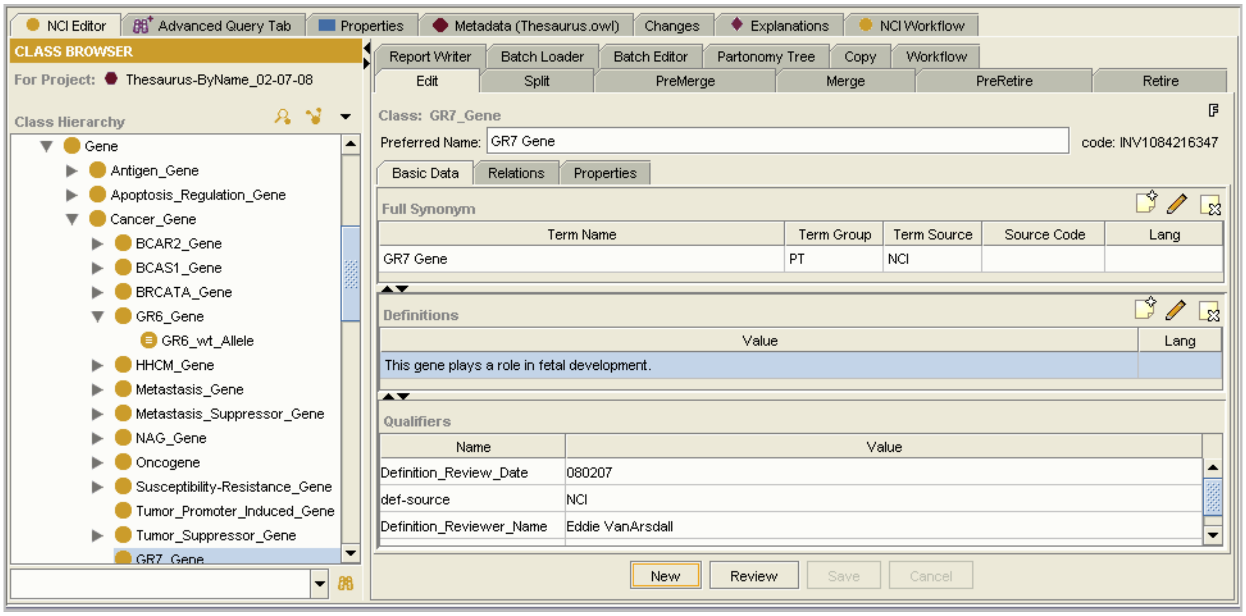
Enter class_name
GR7_Gene

Enter preferred name
GR7 Gene

Enter definition
This gene plays a role in fetal development.

Figure 6.10 Create Subclass window with completed fields

11. Click **OK** to close the Create Subclass window.
12. Verify the following results in the main window:
    - The new class is now highlighted in the Class Hierarchy on the left.
    - The Basic Data subtab (*Figure 6.11* on page 91) shows the following information:
        - The Full Synonym panel shows the *Term Name*, *Term Group*, and *Term Source*.
        - The Definitions panel shows the definition.
        - The Qualifiers panel shows the *Definition_Review_Date*, *def-source*, and *Definition_Reviewer_Name*.



Class: GR7_Gene

Preferred Name: GR7 Gene

code: INV1084216347

Term Name	Term Group	Term Source	Source Code	Lang
GR7 Gene	PT	NCI		

Value	Lang
This gene plays a role in fetal development.	

Name	Value
Definition_Review_Date	080207
def-source	NCI
Definition_Reviewer_Name	Eddie VanArsdall

Figure 6.11 New class information

Treeing a Class

Every class must have a *superclass*, or parent class. Always try to tree a class in the most specific place. For example, a *Helicase* is a type of enzyme, but more specifically, it is a type of hydrolase. It should therefore be treed under the Hydrolase class.

A treed class inherits all of its parents' attributes. Therefore, by inheritance, a Helicase is also a hydrolase, an enzyme, and a protein. If you tree a class in the right place, then you should be able to trace it all the way back to the top of the tree and have each assertion be true. If any assertion is not true (or sometimes not true), then the class is in the wrong place.

Note: As a general rule, keep multiple treeing of a class to a minimum. Whenever possible, choose a single parent and make all other assertions through a role.

Adding a Parent Class

When using Protégé to tree a class, add the class name using the Edit subtab > Relations subtab. The upper right area of the Parent Class panel includes buttons for adding, modifying, and deleting parent classes.

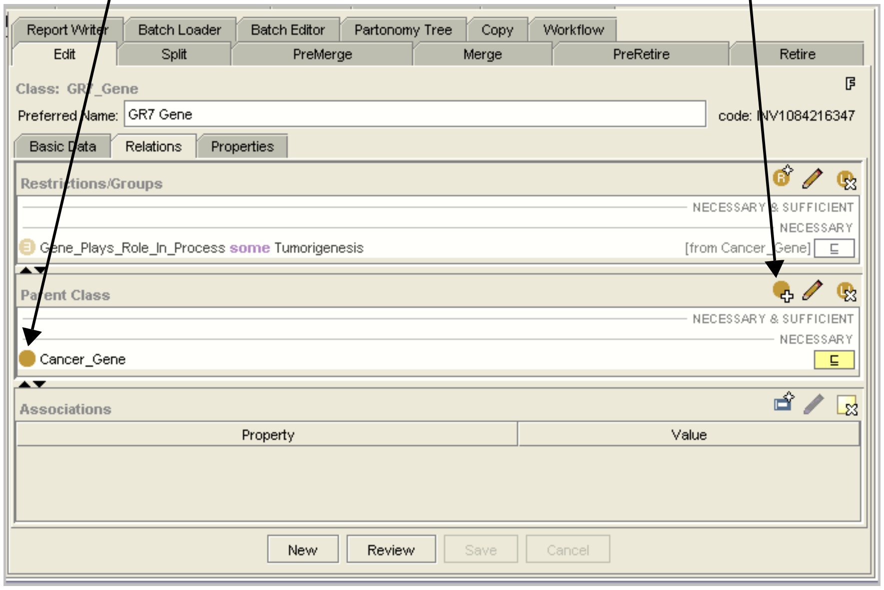
To add a parent class for a specific class, follow these steps:

1. Using the Class Browser, select the class to which you want to add a parent class.
2. Click the **Edit** subtab on the right if it is not already displayed.
3. Click the **Relations** subtab.
4. Select a class in the Parent Class panel.

5. Click the **Add Named Class** button (*Figure 6.12*).

Select a class...

...then click the Add parent class button.



Class: GR7_Gene

Preferred Name: GR7 Gene

code: INV1084216347

Gene_Plays_Role_In_Process some Tumorigenesis [from Cancer_Gene]

Cancer_Gene

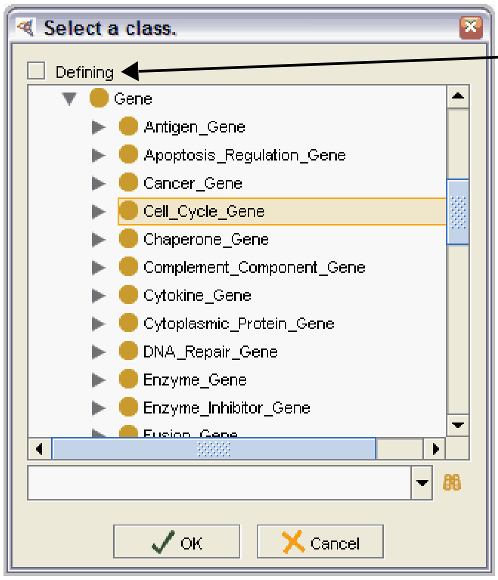
Figure 6.12 Edit subtab - Parent Class panel

The Select a class window opens (*Figure 6.13*).

6. Select or search for a class.

   If you search for a class, the Advanced Query window opens. Select a search result, then click **OK** to close the window and return to the Select a class window.

7. If the selected class is *defining*, check the **Defining** box.



Defining check box

☐ Defining

- Gene
  - Antigen_Gene
  - Apoptosis_Regulation_Gene
  - Cancer_Gene
  - Cell_Cycle_Gene
  - Chaperone_Gene
  - Complement_Component_Gene
  - Cytokine_Gene
  - Cytoplasmic_Protein_Gene
  - DNA_Repair_Gene
  - Enzyme_Gene
  - Enzyme_Inhibitor_Gene
  - Fusion_Gene

Figure 6.13 Select a class window

8. Click **OK** to close the Select a class window.

   The Parent Class panel now shows the new class (*Figure 6.14*).

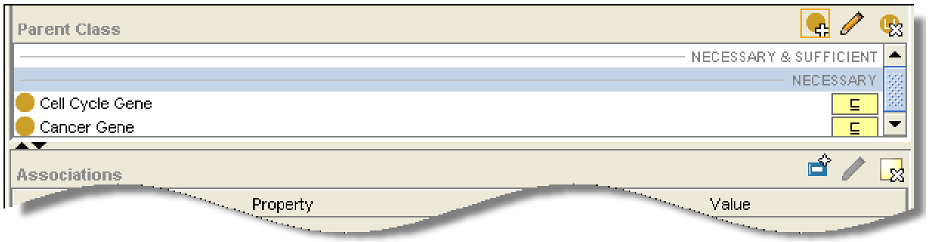


Figure 6.14 Parent Class panel with new class

Tip: (Optional) Click the **Review** button at the bottom of the Edit subtab to review the change in the Review window before saving it. When finished, click the **Close** button to close the window.

9. Click the **Save** button to accept the change.

   A message confirms that the class has been saved successfully.

10. Click **OK** to close the message window.

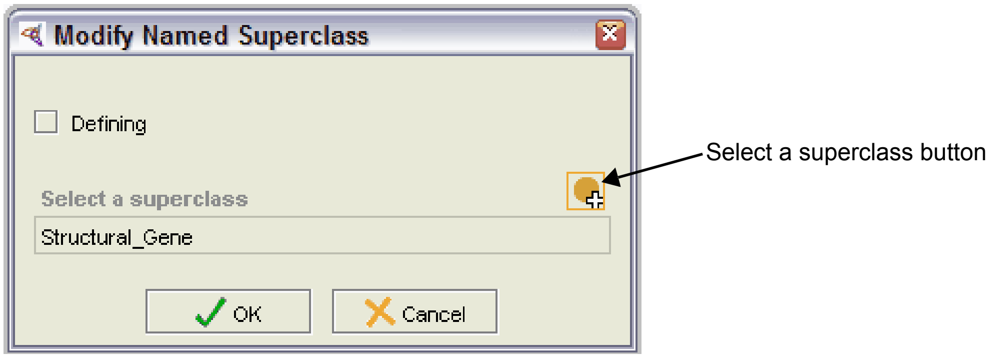
Modifying a Parent Class

To modify a parent class, follow these steps:

1. Using the Class Browser, select the class to be modified.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Relations** subtab.
4. Select the class to be modified in the Parent Class panel.
5. Click the **Edit parent class** button ✎ in the upper right area of the panel.

   The Modify Named Superclass window opens (*Figure 6.15*).

6. Click the **Select a superclass** button ● on the right.

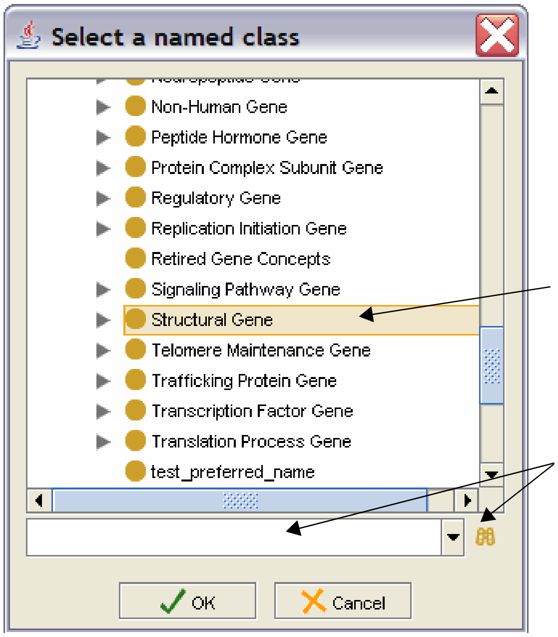


Structural_Gene

Select a superclass button

Figure 6.15 Modify Named Superclass window

7. Select or search for a class.



Select a class...

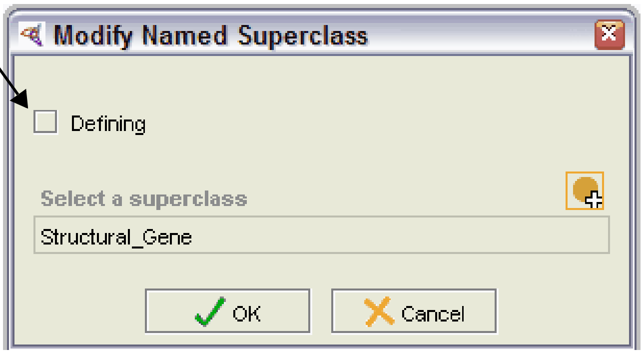
...or search for a class using the search field and button.

Figure 6.16 Select a named class window

If you search for a class, the Advanced Query window opens. Select a search result, then click **OK** to close the window and return to the Select a superclass window.

8. Click **OK** to close the Select a named class window.
9. (Optional) If the new class is a defining class, check the **Defining** box.

Defining check box



Structural_Gene

Figure 6.17 Modified parent class

10. Click **OK** to close the Modify Named Superclass window.

    A message confirms that the class has been saved successfully.

11. Click **OK** to close the message window.

Deleting a Parent Class

To delete a parent class, follow these steps:

1. Using the Class Browser, select the class for which you want to delete a parent class.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Relations** subtab.
4. Select the class to be deleted in the Parent Class panel.

   Note: If only one parent class shows in the list, you cannot delete it.
5. Click the **Delete Selected Row** button in the upper right area of the panel.
6. Click **Yes** when the confirmation message appears.

   The selected class is removed from the Parent Class panel.
7. Click the **Save** button to save the change.

   A message confirms that the class has been saved successfully.
8. Click **OK** to close the message window.

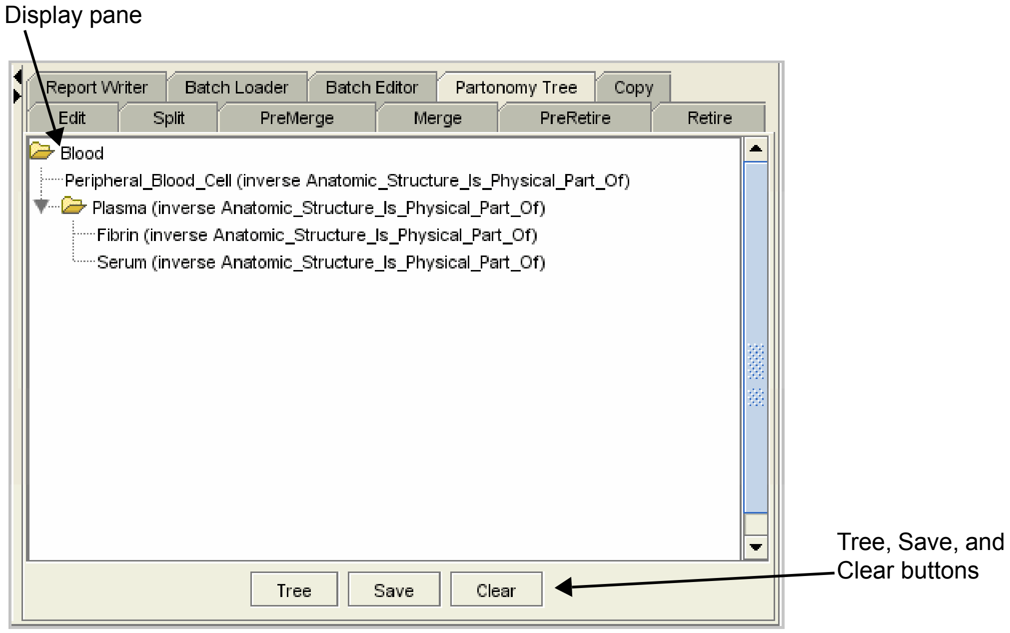
Viewing Classes Connected by part_of Relations

A *partonomy tree* shows classes that are connected by *part_of* relations. This topic explains how to generate a partonomy tree.

- *About the Partonomy Tree Subtab* on this page
- *Generating a Partonomy Tree* on page 97

About the Partonomy Tree Subtab

The Partonomy Tree subtab enables you to select a root class and display it as a partonomy tree.



```
Blood
  Peripheral_Blood_Cell (inverse Anatomic_Structure_Is_Physical_Part_Of)
  Plasma (inverse Anatomic_Structure_Is_Physical_Part_Of)
    Fibrin (inverse Anatomic_Structure_Is_Physical_Part_Of)
    Serum (inverse Anatomic_Structure_Is_Physical_Part_Of)
```

Figure 6.18 Partonomy Tree subtab

Table 6.6 lists the features of the **Partonomy Tree** subtab.

Feature	***Description***
Display pane	Shows a partonomy tree generated from a selected root class.
Tree button	Opens the Select Transitive Properties window, in which you can select a restriction for a selected root class, or change the root class.
Save button	Saves the partonomy tree as an ASCII file.
Clear button	Clears the display pane.

Table 6.6 Partonomy Tree subtab features

Generating a Partonomy Tree

To generate a partonomy tree from a root class, follow these steps:

1. Using the Class Browser, select a root class.
2. Click the **Partonomy Tree** subtab.
3. Click the **Tree** button at the bottom of the subtab (*Figure 6.19*).

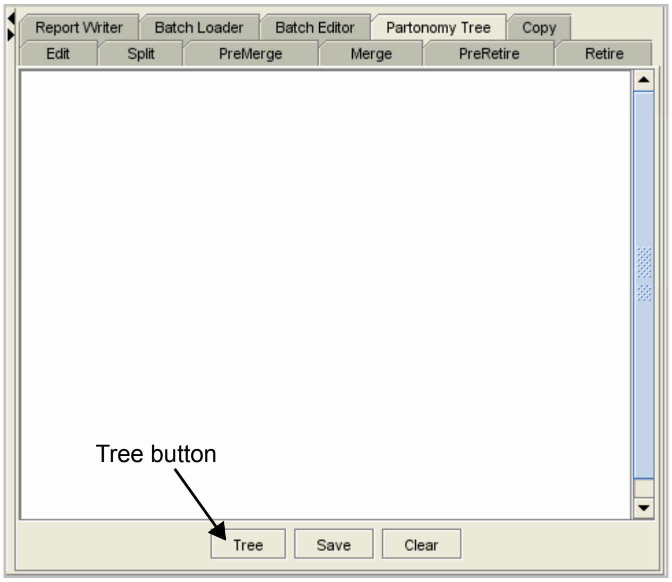
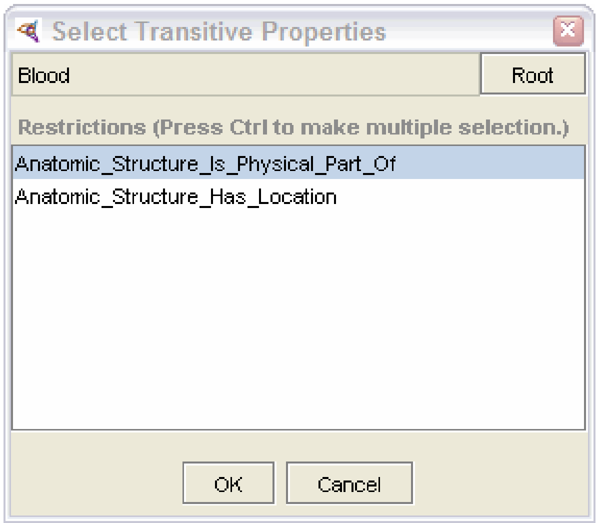


Figure 6.19 Partonomy Tree subtab

The Select Transitive Properties window opens (*Figure 6.20*).

4. Select one or more restriction names.

Tip: To select multiple restrictions, press and hold the CTRL key while clicking each item.



Blood

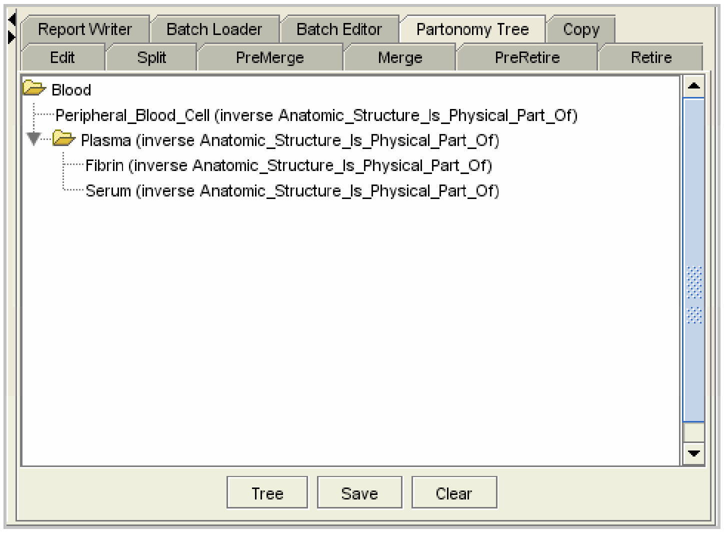
Restrictions (Press Ctrl to make multiple selection.)

Anatomic_Structure_Is_Physical_Part_Of
Anatomic_Structure_Has_Location

Figure 6.20 Select Transitive Properties window

5. Click **OK** to close the window and generate the tree.

   The partonomy tree appears in the tree view area of the subtab (*Figure 6.21*).



```
Blood
  Peripheral_Blood_Cell (inverse Anatomic_Structure_Is_Physical_Part_Of)
  Plasma (inverse Anatomic_Structure_Is_Physical_Part_Of)
    Fibrin (inverse Anatomic_Structure_Is_Physical_Part_Of)
    Serum (inverse Anatomic_Structure_Is_Physical_Part_Of)
```

Figure 6.21 Partonomy tree for Blood class

6. (Optional) Click the **Save** button to save the tree to an ASCII file.

   Note: When prompted to name the new file, append the `.txt` extension to the file name.

CHAPTER
7
WORKING WITH PROPERTIES

This chapter explains how to use NCI Protégé to create and manage properties.

Topics in this Chapter

Asserting Annotation Properties

As you work with class properties in NCI Protégé, follow these guidelines:

- *Add* properties to a class when you want to add new full synonyms, definitions, term groups, or term sources.
- *Modify* properties when you need to do any of the following:
  - Add qualifiers
  - Correct spelling errors or punctuation
  - Change singular to plural, or vice-versa
  - Change the term group or term source.

  Note: Ensure that you modify only properties that have NCI as the term source. You cannot modify NCI-GLOSS definitions, but you can delete them if they are inaccurate or if they are duplicates.
- *Delete* properties when you spot duplicates or other properties created in error.

Note: In this chapter, you will work with the Edit subtab and its three subtabs: Basic Data, Relations, and Properties. For an introduction to these subtabs, see *Examining Class Information* on page 82

Adding a Full Synonym

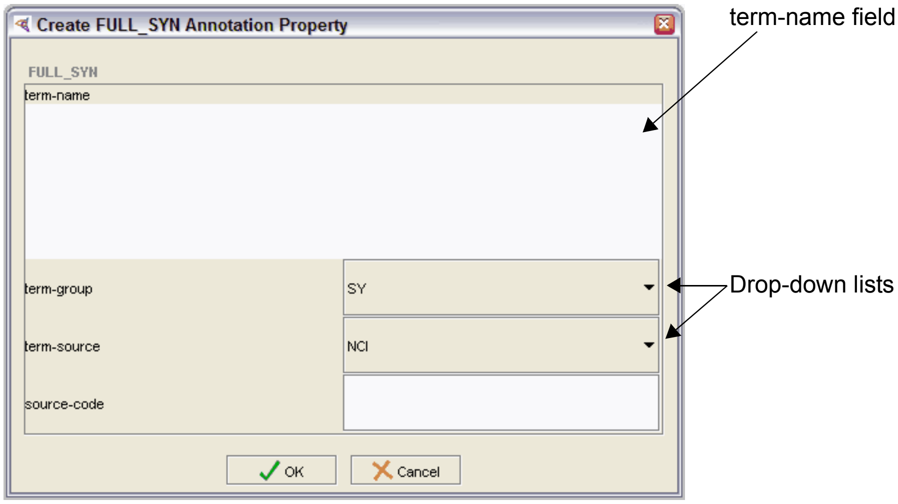
A fully qualified synonym contains the term name, term group, term source, and an optional source code if appropriate.

To add a full synonym, follow these steps:

1. Using the Class Browser, select the class requiring a new synonym.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Basic Data** subtab to view basic properties for the selected class.
4. Click the **Add full synonym** button in the upper right area of the Full Synonym panel.

   The Create FULL_SYN Annotation Property window opens (*Figure 7.1*).
5. Enter a value in the **term-name** field.
6. Ensure that the term group is **SY** and that the term source is **NCI**.

   If you need to change either value, select it from the appropriate drop-down list.



Create FULL_SYN Annotation Property

FULL_SYN

term-name

term-group: SY

term-source: NCI

source-code

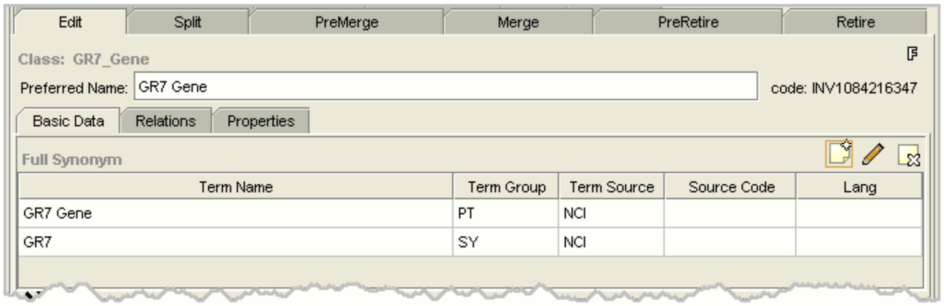
term-name field

Drop-down lists

Figure 7.1 Create FULL_SYN Annotation Property window

7. Click **OK** to close the window.

The new synonym now appears in the Full Synonym list.



Class: GR7_Gene

Preferred Name: GR7 Gene

code: INV1084216347

Full Synonym

Term Name	Term Group	Term Source	Source Code	Lang
GR7 Gene	PT	NCI		
GR7	SY	NCI		

Figure 7.2 New synonym

8. Click the **Save** button to save the change.

   A message confirms that the class has been saved successfully.

9. Click **OK** to close the message window.

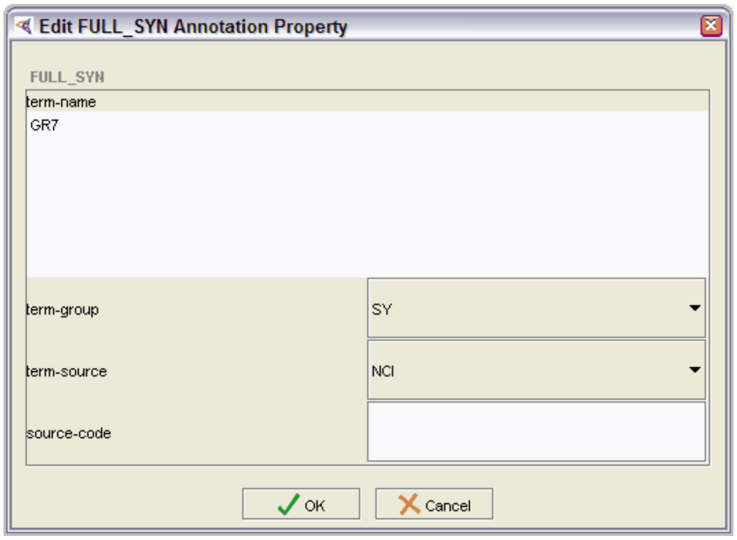
Modifying a Full Synonym

To modify a full synonym, follow these steps:

1. Using the Class Browser, select the class to be modified.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Basic Data** subtab to view basic properties for the selected class.
4. Select the **Full Synonym** to be edited.
5. Click the **Edit full synonym** button ✎ in the upper right area of the Full Synonym panel.

   The Edit FULL_SYN Annotation Property window opens (*Figure 7.3*).

6. Change information as necessary (for example, *term name* or *term group*).



FULL_SYN

term-name: GR7

term-group: SY

term-source: NCI

source-code:

Figure 7.3 Edit FULL_SYN Annotation Property window

7. Click **OK** to accept the change and close the window.
8. Click the **Save** button to save the change.

   A message confirms that the class has been saved successfully.

   Note: If you change the *term-group* property to PT and a preferred term already exists, an error message appears. If this happens, repeat this procedure and select another term group.

9. Click **OK** to close the message window.

Deleting a Full Synonym

To delete a full synonym, follow these steps:

1. Using the Class Browser, select the class from which you want to delete a full synonym.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Basic Data** subtab to view basic properties for the selected class.
4. Select the property to be deleted in the Full Synonym panel.
5. Click the **Delete full synonym** button in the upper right area of the panel.
6. When the confirmation message appears, click **Yes**.

   The selected property is removed from the Full Synonym panel.

   Tip: If you deleted a property by mistake, click the **Cancel** button to restore the property.

7. Click the **Save** button to save the change.

   A message confirms that the class has been saved successfully.

8. Click **OK** to close the message window.

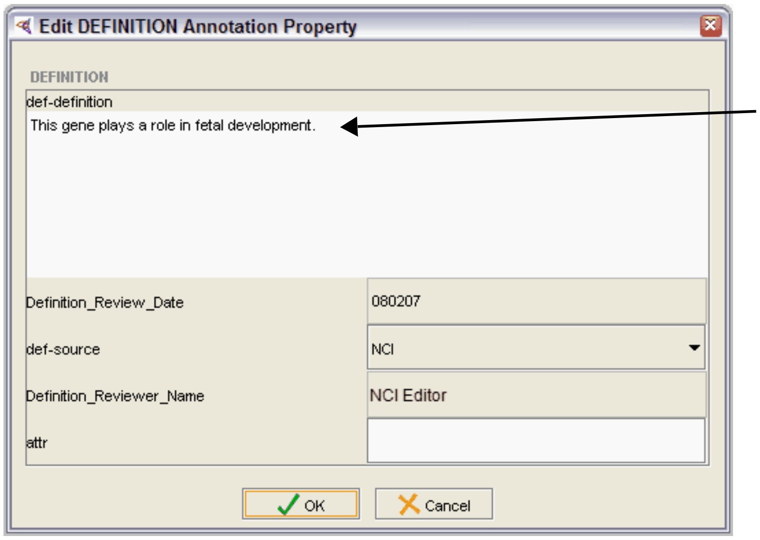
Modifying a Definition

To modify a class definition or qualifier, follow these steps:

1. Using the Class Browser, select the class to be modified.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Basic Data** subtab.
4. If more than one definition exists, select the definition to be modified. Otherwise, the definition should already be selected.
5. Click the **Edit definition** button in the upper right area of the Definitions panel.

   The Edit DEFINITION Annotation Property window opens (*Figure 7.4* on page 103).

6. Edit the text in the **def-definition** field as desired.



DEFINITION

def-definition

This gene plays a role in fetal development.

Definition_Review_Date: 080207

def-source: NCI

Definition_Reviewer_Name: NCI Editor

attr:

def-definition field

Figure 7.4 Edit DEFINITION Annotation Property window

7. Click **OK** to close the window.
8. Click the **Save** button to save the change.

   A message confirms that the class has been saved successfully.
9. Click **OK** to close the message window.

Asserting Relations

The following topics explain how to work with restrictions and role groups. For more information about roles and role groups, see *Chapter 2, Description Logic and the NCI Thesaurus Semantic Model,* on page 13.

- *Adding a Restriction* on this page
- *Modifying a Restriction*
- *Deleting a Restriction*
- *Adding a Role Group*

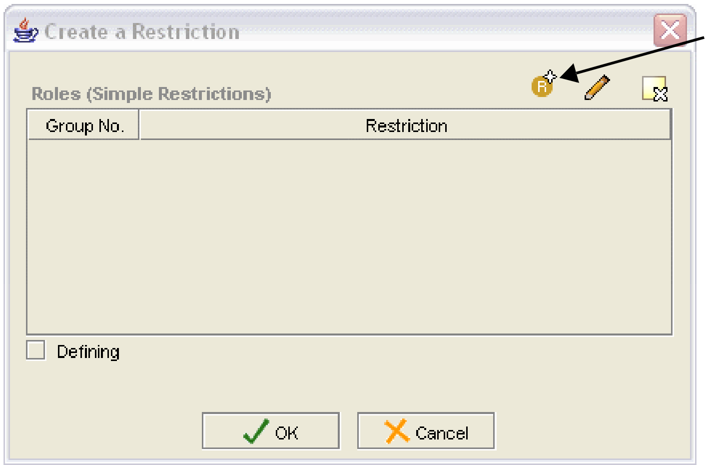
Adding a Restriction

To add a simple restriction to a class, follow these steps:

1. Using the Class Browser, select the class to be modified.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Relations** subtab.
4. Click the **Add a restriction/group** button in the upper right area of the Restrictions/Groups panel.

   The Create a Restriction window opens (*Figure 7.5*).

5. Click the **Create a Role** button, the first of the three buttons in the right side of the window.



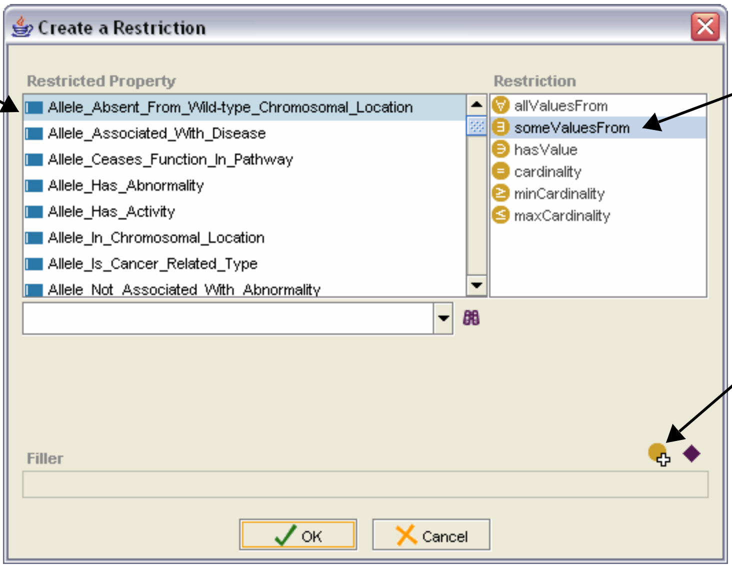
Create a Role button

Figure 7.5 Create a Restriction window (initial)

The Create a Restriction window now displays two lists: Restricted Properties and Restrictions (*Figure 7.6*).

6. Select a property in the **Restricted Property** list.
7. Select a modifier in the **Restriction** list (for example, `someValuesFrom`).
8. Click the **Select a Named Class (filler)** button in the lower right area of the window (*Figure 7.6*).

Restricted Property list



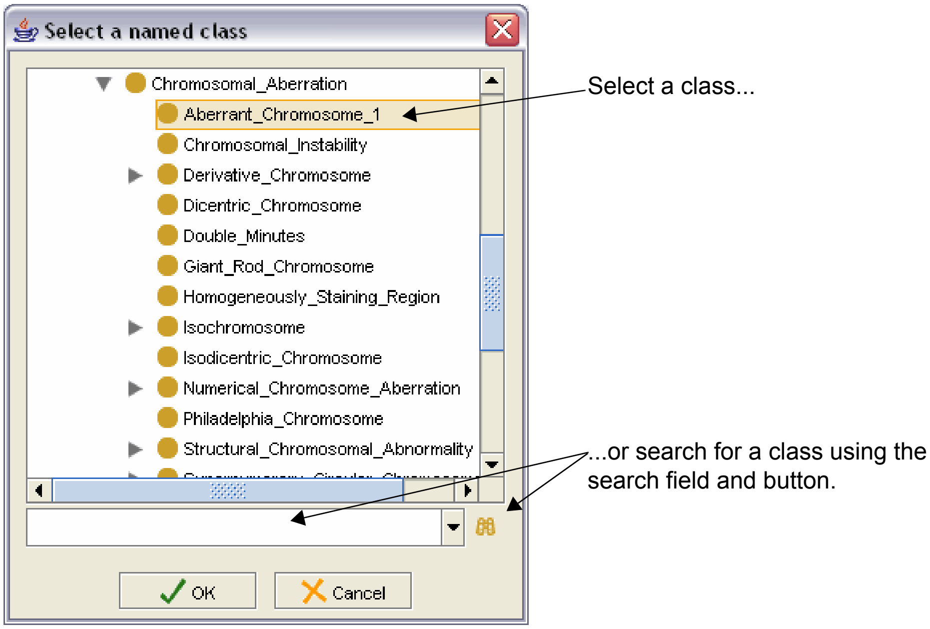
Restriction list

Select a Named Class (filler) button

Figure 7.6 Create a Restriction window (second of two windows)

The Select a named class window opens (*Figure 7.7*).

9. Select or search for a class.



Select a class...

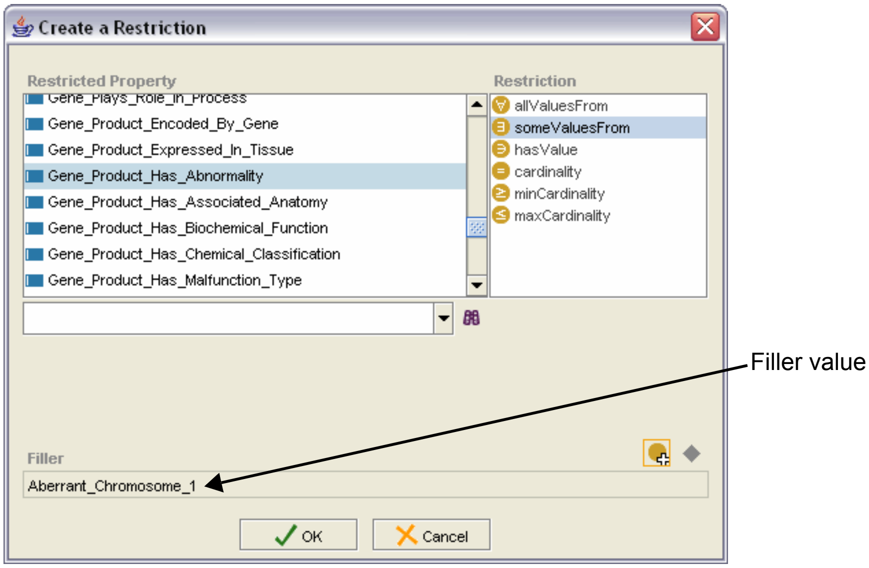
...or search for a class using the search field and button.

Figure 7.7 Select a named class window

Note: If you search for a class, the Advanced Query window opens. Select a search result, then click **OK** to close the window and return to the Select a named class window.

10. Click **OK** to close the Select a named class window.

In the Create a Restriction window, the class that you selected in Step 9 now shows as the filler value (*Figure 7.8*).



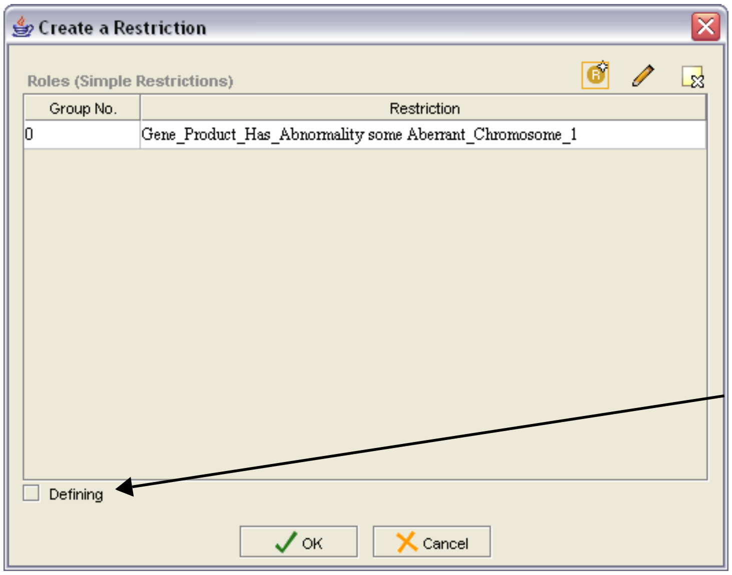
Filler
Aberrant_Chromosome_1

Filler value

Figure 7.8 Create a Restriction window with Filler value

11. Click **OK** to return to the original view of the Create a Restriction window.

    The new restriction appears in the Roles (Simple Restrictions) list (*Figure 7.9*).



Group No.	Restriction
0	Gene_Product_Has_Abnormality some Aberrant_Chromosome_1

☐ Defining

Figure 7.9 Newly added restriction

12. (Optional) If the current role is a defining role, check the **Defining** box in the lower left of the window.
13. Click **OK**.

    The new restriction now appears in the **Restrictions/Groups** list.

14. Click the **Save** button to save the change.

    A message confirms that the class has been saved successfully.

15. Click **OK** to close the message window.

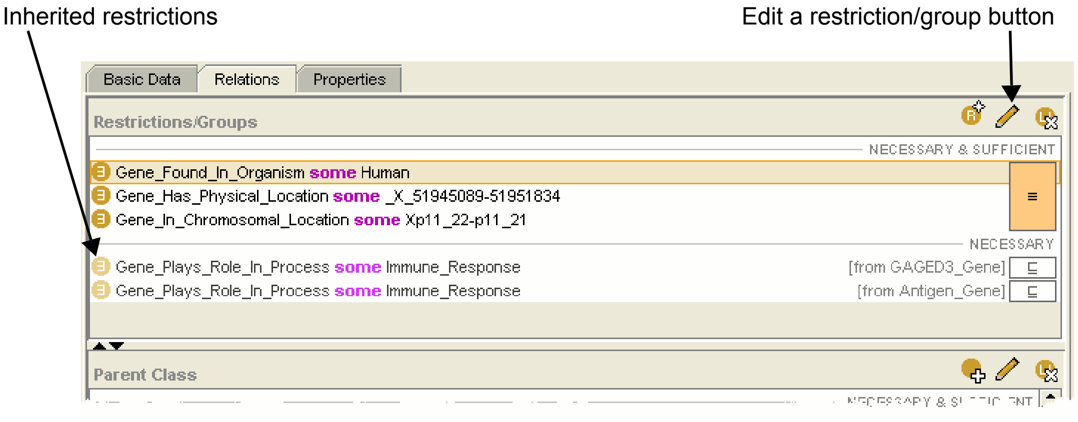
Modifying a Restriction

To modify a restriction, follow these steps:

1. Using the Class Browser, select the class to be modified.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Relations** subtab.
4. Select the restriction to be edited on the Restrictions/Groups panel.

    Note: Inherited restrictions appear in the lower area of the panel. Even though you can select them, you cannot edit them. If you try, the following message appears: *Cannot modify inherited restriction*.

5. Click the **Edit a restriction/group** button in the upper right area of the panel (*Figure 7.10*).



Gene_Found_In_Organism some Human
Gene_Has_Physical_Location some _X_51945089-51951834
Gene_In_Chromosomal_Location some Xp11_22-p11_21

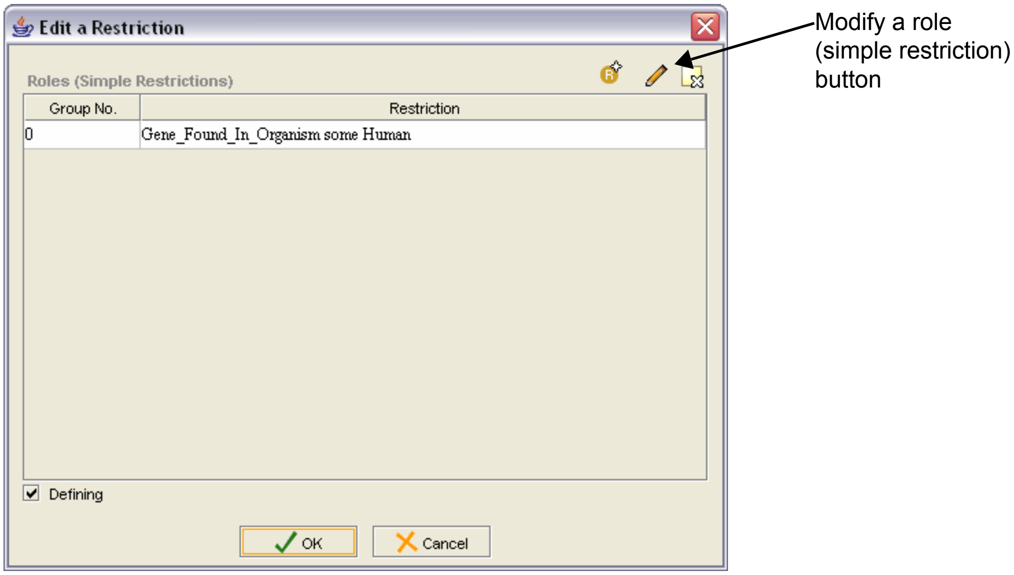
Gene_Plays_Role_In_Process some Immune_Response [from GAGED3_Gene]
Gene_Plays_Role_In_Process some Immune_Response [from Antigen_Gene]

Figure 7.10 Restrictions/Groups panel with inherited restrictions

The Edit a Restriction window opens (*Figure 7.11*).

6. Select the restriction to be modified, even if only one restriction appears in the window.

7. Click the **Modify a role (simple restriction)** button in the upper right area of the window (*Figure 7.11*).



Group No.	Restriction
0	Gene_Found_In_Organism some Human

☑ Defining

Figure 7.11 Edit a Restriction window

The Create a Restriction window now displays two lists: Restricted Properties and Restrictions (*Figure 7.12*).

8. Select a property in the **Restricted Property** list.

9. Select a modifier in the **Restriction** list (for example, `someValuesFrom`).

10. Click the **Select a Named Class (filler)** button in the lower right area of the window (*Figure 7.12*).

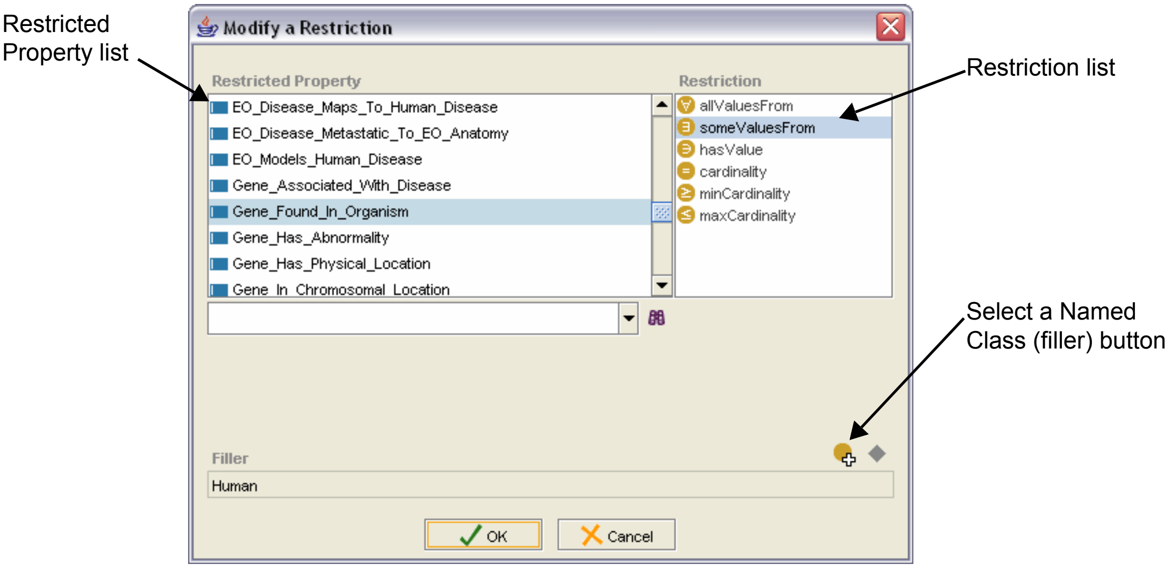


Figure 7.12 Modify a Restriction window

The Select a named class window opens (*Figure 7.13*).

11. Select or search for a class.

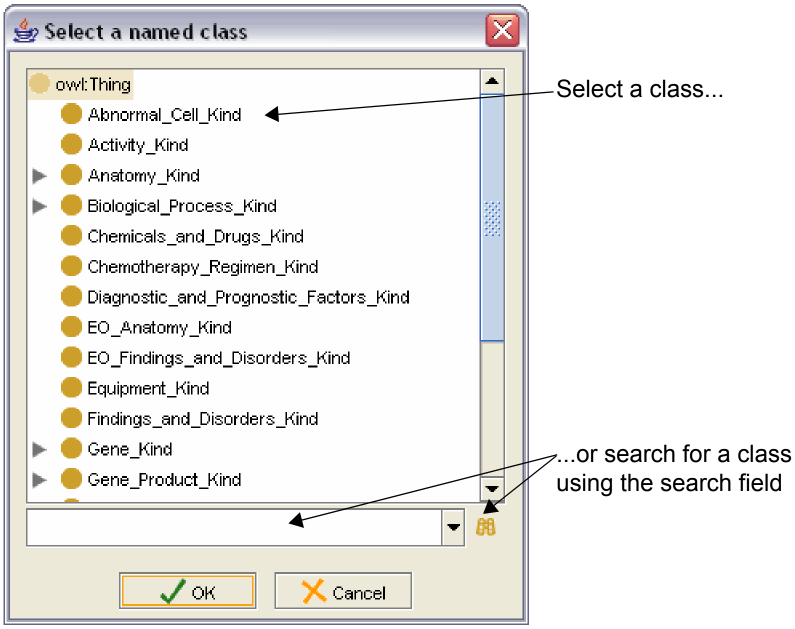


Figure 7.13 Select a named class window

If you search for a class, the Advanced Query window opens. Select a search result, then click **OK** to close the window and return to the Select a superclass window.

12. Click **OK** to close the Select a named class window.

In the Create a Restriction window, the class that you selected in Step 11 now shows as the filler value (*Figure 7.14*).

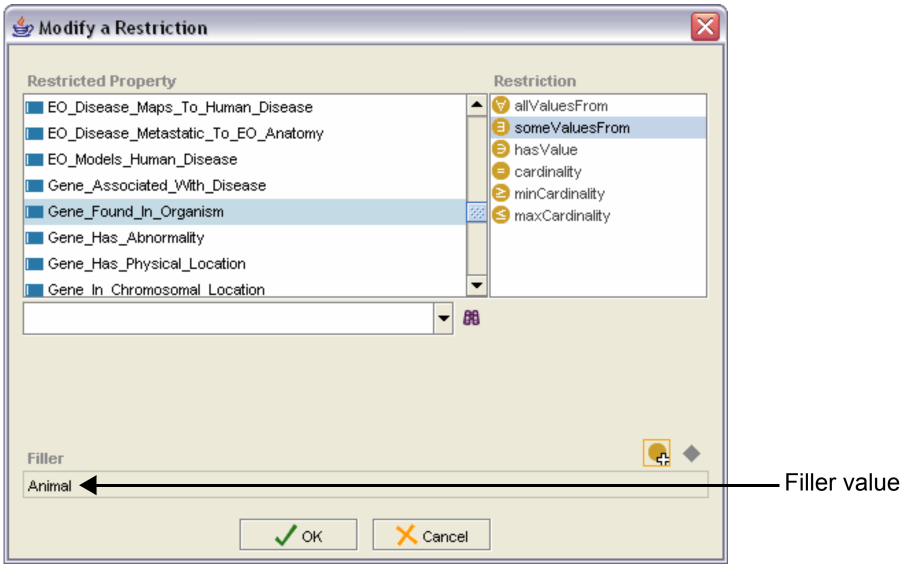


Figure 7.14 Modify a Restriction window with new Filler value

13. Click **OK** to return to the Edit a Restriction window.

 The new restriction now appears in the original window.

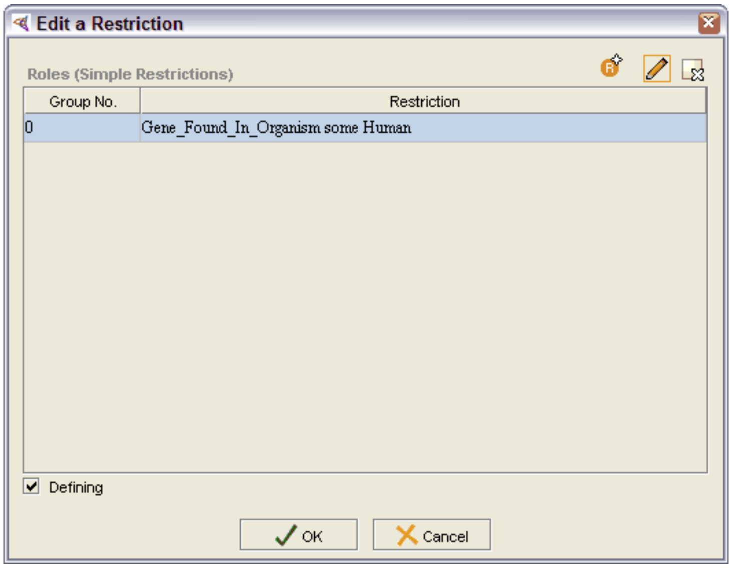


Figure 7.15 Newly modified restriction

14. Click **OK** to close the Edit a Restriction window.
15. Click the **Save** button to save the change.

 A message confirms that the class has been saved successfully.

16. Click **OK** to close the message window.

Deleting a Restriction

To delete a restriction, follow these steps:

1. Using the Class Browser, select the class from which you want to delete a restriction.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Relations** subtab.
4. Select the restriction to be deleted on the Restrictions panel.
5. Click the **Delete selected row** button in the upper right area of the panel.
6. When the confirmation message appears, click **Yes**.

   The selected restriction is removed from the Restrictions panel. If you deleted a restriction by mistake, click the **Cancel** button to restore the property.
7. Click the **Save** button to save the change.
8. When the confirmation message appears, click **OK** to close the message window.

Adding a Role Group

NCI Protégé does not directly support intersections or unions. However, both are used in the creation of class expressions called *role groups*.

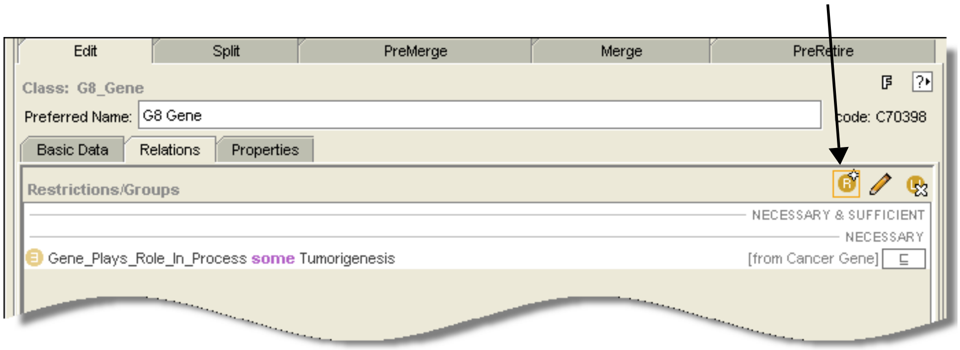
To add a role group, perform the following two procedures in the following order. For more information about role groups, see *OWL Class Descriptions and Anonymous Classes* on page 18.

Adding Multiple Restrictions

A role group requires multiple restrictions. To add the restrictions, follow these steps:

1. Using the Class Browser, select the class to which you want to add a role group.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Relations** subtab.
4. Click the **Create a restriction** button in the upper right area of the Restrictions/Groups panel (*Figure 7.16*).

Create a restriction button



Class: G8_Gene

Preferred Name: G8 Gene

Code: C70398

Gene_Plays_Role_In_Process some Tumorigenesis [from Cancer Gene] ⊑

Figure 7.16 Edit sub > Relations subtab > Restrictions panel

The Create a Restriction window opens (*Figure 7.17*).

5. Click the **Create a Role** button (the first of the three buttons in the right side of the window).

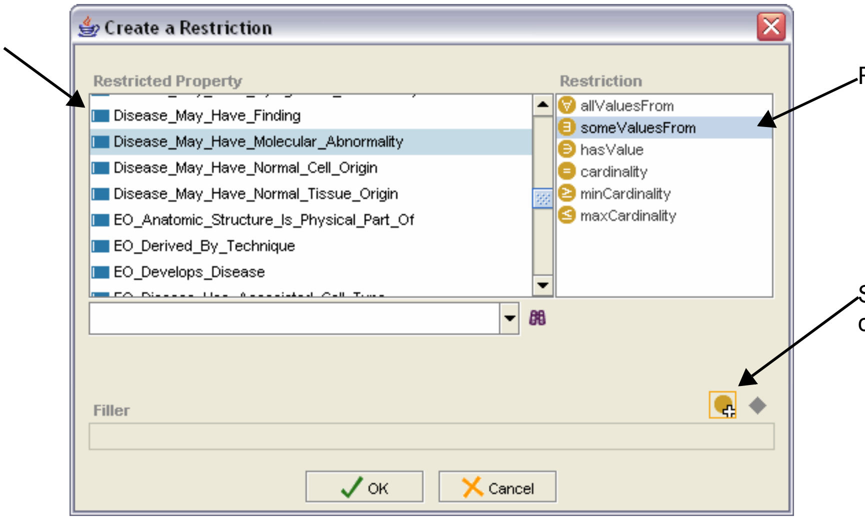
Create a Role button

Figure 7.17 Create a Restriction window (first of two windows)

The Create a Restriction window now displays two lists: Restricted Properties and Restrictions.

6. Select a property in the **Restricted Property** list.
7. Select a modifier in the **Restriction** list (for example, `someValuesFrom`).
8. Click the **Select a Named Class** (filler) button in the lower right area of the window (*Figure 7.18*).

Restricted Property list



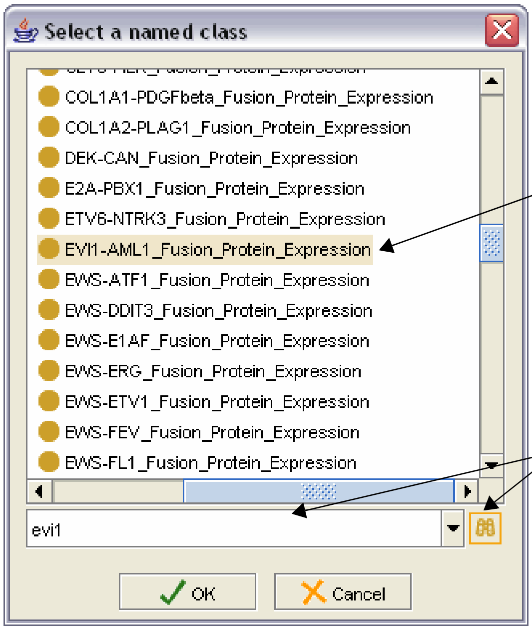
Restriction list

Select a named class (filler) button

Figure 7.18 Create a Restriction window (second of two windows)

The Select a named class window opens (*Figure 7.19*).

9. Select or search for a class (*Figure 7.19*).



evi1

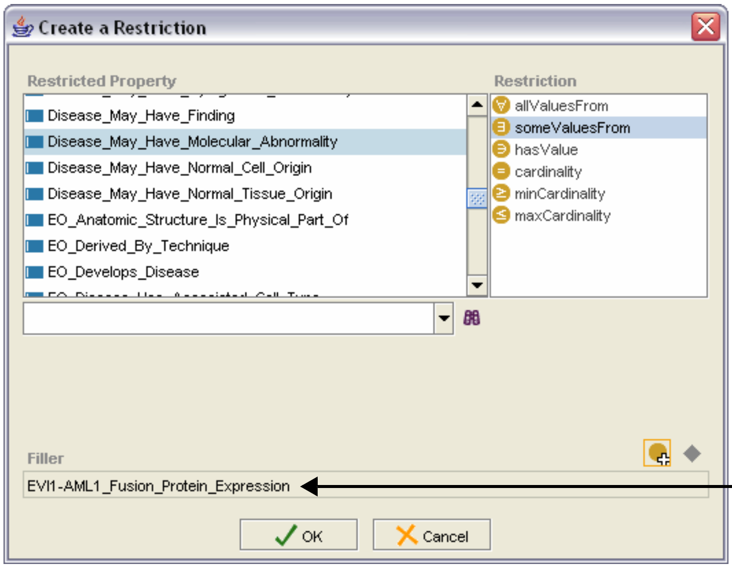
Select a class...

...or search for a class using the search field and button.

Figure 7.19 Select a named class window

If you search for a class, the Advanced Query window opens. Select a search result, then click **OK** to close the window and return to the Select a superclass window.

In the Create a Restriction window, the class that you selected in Step 11 now shows as the filler value (*Figure 7.20*).



Filler

EVI1-AML1_Fusion_Protein_Expression

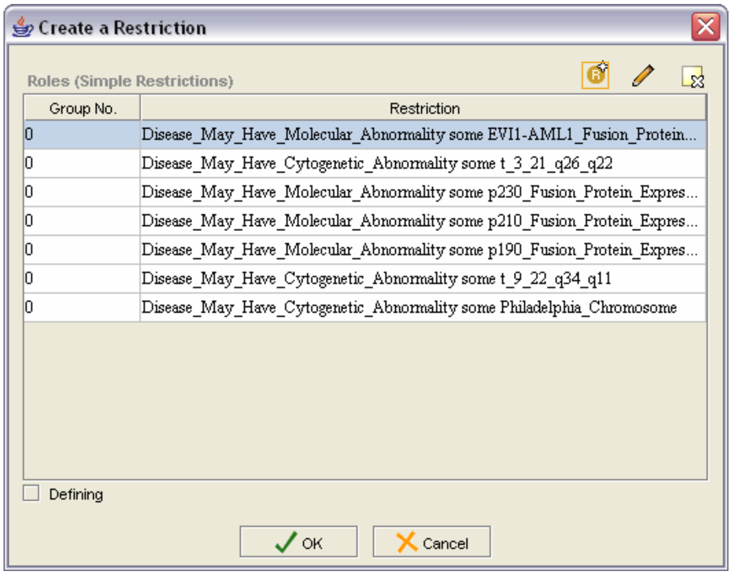
Filler value

Figure 7.20 Create a Restriction window with Filler value

10. Repeat Steps 6 through 9 to add more restrictions.

11. Once you are finished, click **OK** to return to the original window.

The window now shows a group of restrictions.



Group No.	Restriction
0	Disease_May_Have_Molecular_Abnormality some EVI1-AML1_Fusion_Protein...
0	Disease_May_Have_Cytogenetic_Abnormality some t_3_21_q26_q22
0	Disease_May_Have_Molecular_Abnormality some p230_Fusion_Protein_Expres...
0	Disease_May_Have_Molecular_Abnormality some p210_Fusion_Protein_Expres...
0	Disease_May_Have_Molecular_Abnormality some p190_Fusion_Protein_Expres...
0	Disease_May_Have_Cytogenetic_Abnormality some t_9_22_q34_q11
0	Disease_May_Have_Cytogenetic_Abnormality some Philadelphia_Chromosome

Figure 7.21 Multiple restrictions

12. Leave the Create a Restriction window open and continue to the next procedure on page 114.

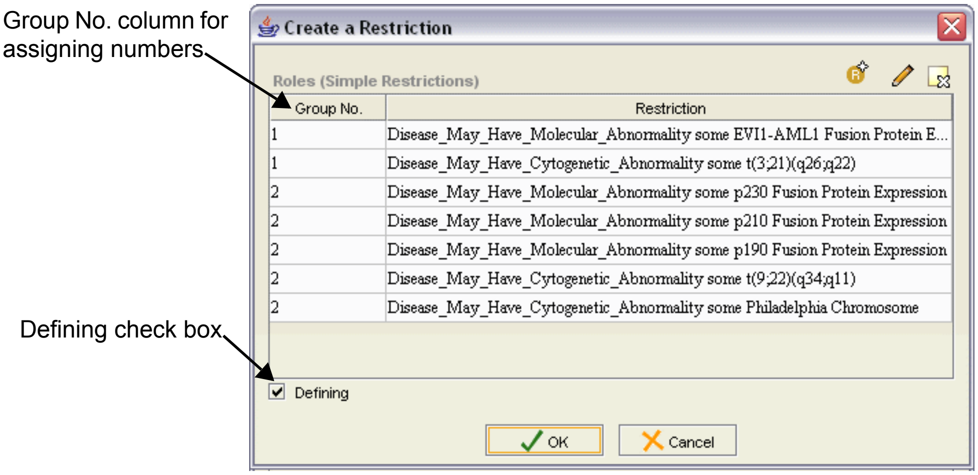
Building a Role Group Expression

Prerequisite procedure: *Adding Multiple Restrictions*

Caution: If you assign a zero (0) instead of a one (1) in the first two steps, you will build a class expression—not a role group expression.

To build a role group expression from a group of restrictions, follow these steps:

1. Starting with the first item in the list, double-click the number on the left, under the heading **Group No.** (*Figure 7.22*).
2. Type a new number, then click elsewhere to deselect the field.
3. Repeat Steps 1 and 2 for each item in the group.
4. (Optional) If applicable, check the **Defining** box in the lower left of the window.

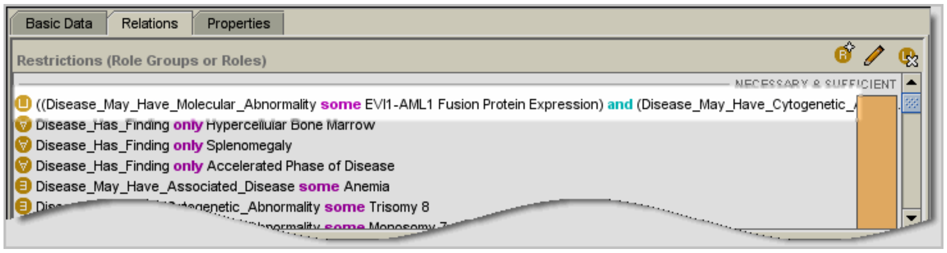


Group No.	Restriction
1	Disease_May_Have_Molecular_Abnormality some EVI1-AML1 Fusion Protein E...
1	Disease_May_Have_Cytogenetic_Abnormality some t(3;21)(q26;q22)
2	Disease_May_Have_Molecular_Abnormality some p230 Fusion Protein Expression
2	Disease_May_Have_Molecular_Abnormality some p210 Fusion Protein Expression
2	Disease_May_Have_Molecular_Abnormality some p190 Fusion Protein Expression
2	Disease_May_Have_Cytogenetic_Abnormality some t(9;22)(q34;q11)
2	Disease_May_Have_Cytogenetic_Abnormality some Philadelphia Chromosome

Figure 7.22 Role group example

5. Click **OK**.

   The new role group now appears in the Relations panel. It is easily identifiable by the length of the expression (*Figure 7.23*).



((Disease_May_Have_Molecular_Abnormality some EVI1-AML1 Fusion Protein Expression) and (Disease_May_Have_Cytogenetic_
Disease_Has_Finding only Hypercellular Bone Marrow
Disease_Has_Finding only Splenomegaly
Disease_Has_Finding only Accelerated Phase of Disease
Disease_May_Have_Associated_Disease some Anemia
genetic_Abnormality some Trisomy 8

Figure 7.23 Role group expression

Tip: To edit the role group expression, select the expression, then right-click and select the appropriate command from the shortcut menu.

6. Click the **Save** button to save the change, then click **OK** to close the save confirmation message window.

Adding an Association

An association is a non-inheriting relationship between two named classes. An example of such a relationship is *Has_Salt_Form*. OWL represents an association as an annotation property of object type.

The following example adds the association *Has_Target* to the concept *Phagocytosis*.

To add an association, follow these steps:

1. Using the Class Browser, follow this path:

   Biological Process Kind > **Biological Process** > **Cellular Process** > **Cell Defense Process** > **Phagocytosis**

2. Click the **Edit** subtab if it is not already displayed.

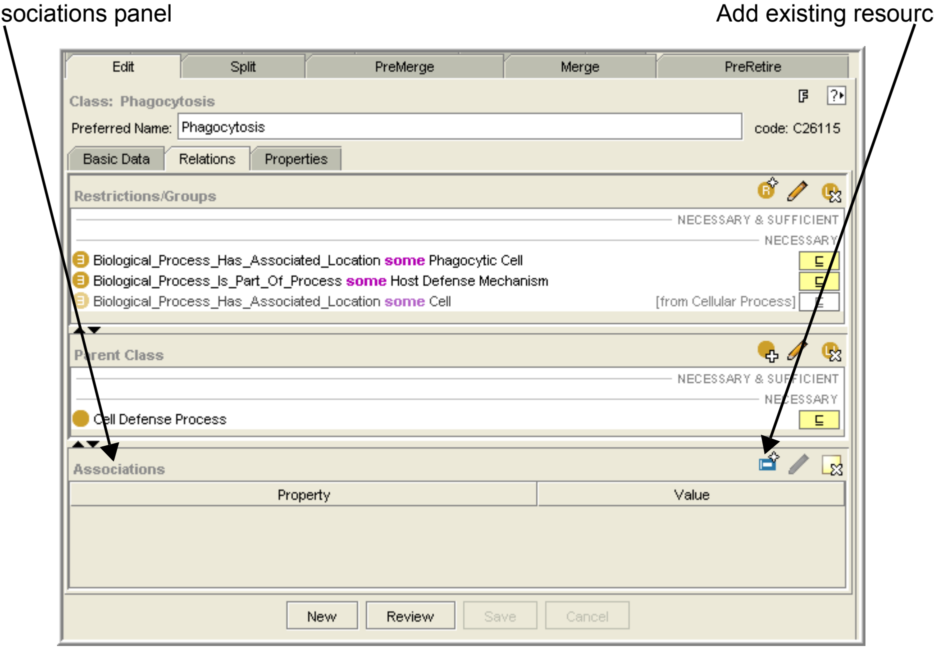
   The **Basic Data** subtab displays information for the *Phagocytosis* class.

3. Click the **Relations** subtab.

4. Click the **Add existing resource as value** button in the upper right area of the Associations panel (*Figure 7.24*).

Associations panel

Add existing resource button



Class: Phagocytosis

Preferred Name: Phagocytosis

code: C26115

Biological_Process_Has_Associated_Location some Phagocytic Cell

Biological_Process_Is_Part_Of_Process some Host Defense Mechanism

Biological_Process_Has_Associated_Location some Cell [from Cellular Process]

Cell Defense Process

Figure 7.24 Relations subtab for Phagocytosis

The Add an Object-Valued Property window opens (*Figure 7.25* on page 116).

5. Click the **Select a property** button in the upper right area.

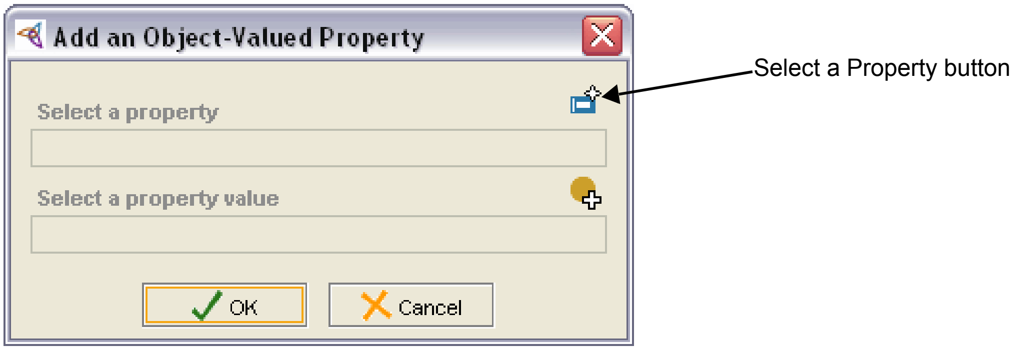


Figure 7.25 Add an Object-Valued Property window

The Select a property window opens (*Figure 7.26*).

6. Select or search for the property *Has_Target*.

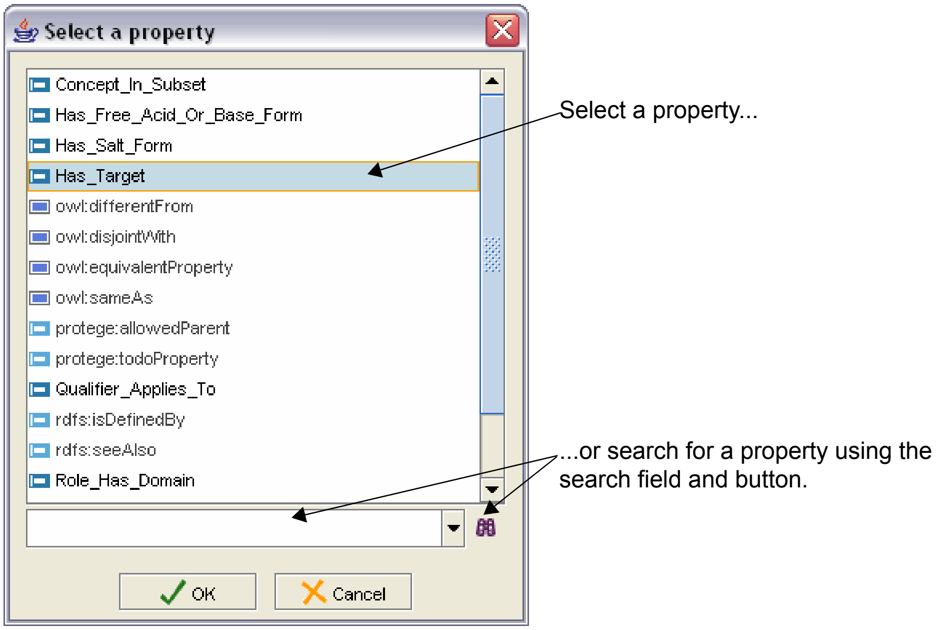
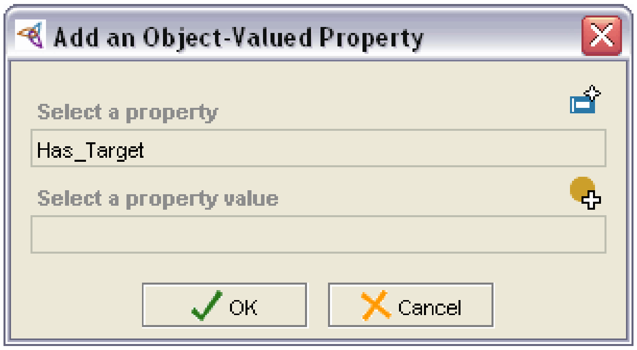


Figure 7.26 Select a Property window

If you search for a class, the Advanced Query window opens. Select a search result, then click **OK** to close the window and return to the Select a superclass window

7. Click **OK** to close the Select a Property window.

In the Add an Object-Valued Property window, the selected value now appears in the Select a property field (*Figure 7.27* on page 117).



Select a property: Has_Target

Select a property value:

Figure 7.27 Select a property value added

8. Click the **Select a property value** button.

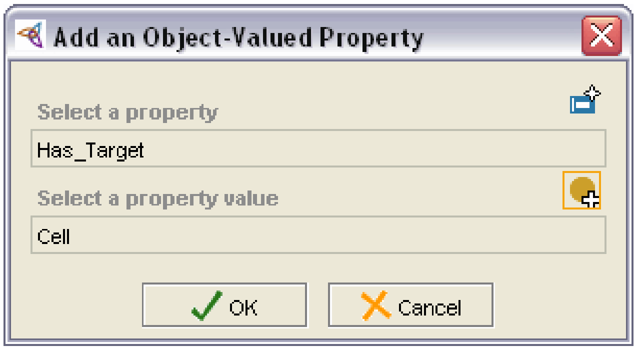
   The Select a property value window opens.

9. Follow this path to select the **Cell** property value:

   Anatomy_Kind > **Anatomic Structure, System, or Substance** > **Microanatomic Structure** > **Cell**

10. Click **OK**.

    The Add an Object-Valued Property window now shows the value *Cell* in the Select a property value field (*Figure 7.28*).



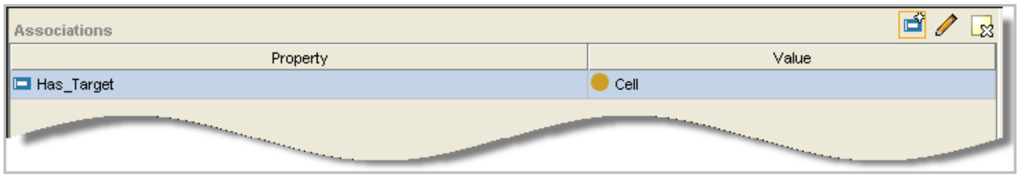
Select a property: Has_Target

Select a property value: Cell

Figure 7.28 Add an Object-Valued Property window with stored values

11. Click **OK** to close the Add an Object-Valued Property window.

    The Associations panel now displays the *Has_Target* property and the *Cell* property value (*Figure 7.29*).



Property	Value
Has_Target	Cell

Figure 7.29 Associations panel

Note: If you add a new subclass for *Phagocytosis*, the association *Has_Target* is not inherited by the subclass, but the restrictions are inherited.

12. Click the **Save** button to save the change.

    A message confirms that the class has been saved successfully.

13. Click **OK** to close the message window.

Editing Two Classes at the Same Time

Sometimes you may want to simultaneously edit two classes. For example, you may want to copy a property from one class to another. Although the Copy subtab is used for this procedure, copying (cloning) a class is a different procedure. For more information, see *Cloning a Class* on page 130.

To edit two classes simultaneously, follow these steps:

1. In the main Protégé window, click the **Copy** subtab.
2. Drag a class into the upper pane.

   If you drag a class into the lower pane and the upper pane is empty, Protégé automatically places the class in the upper pane.

3. Drag a class into the lower pane.
4. Use the right-click menu to add or modify properties:
   - Right-click the **class name** to do any of the following:
     - Add a property
     - Create a restriction
     - Add an association
     - Add a parent
     - Create a workflow task (workflow managers only)
   - Right-click a **property** to do any of the following:
     - Delete the property
     - Modify the property
     - Create a workflow task (workflow managers only)
     - Copy the property
     - Paste a copied property
5. Click the **Save** button to save the change.
6. Click **OK** to close the save confirmation message.

   Note: If you need to clear the upper and lower panes and edit more classes, *save your edits*, then click the **Clear** button below the Copy subtab.

CHAPTER

8

SPLITTING, MERGING, AND CLONING CLASSES

Sometimes you need to create new classes from existing classes or combine two classes to form one class. This chapter explains the techniques of splitting, merging, and cloning classes.

Topics in this Chapter

Splitting a Class

The following topics are relevant to splitting classes:

When to Consider Splitting a Class

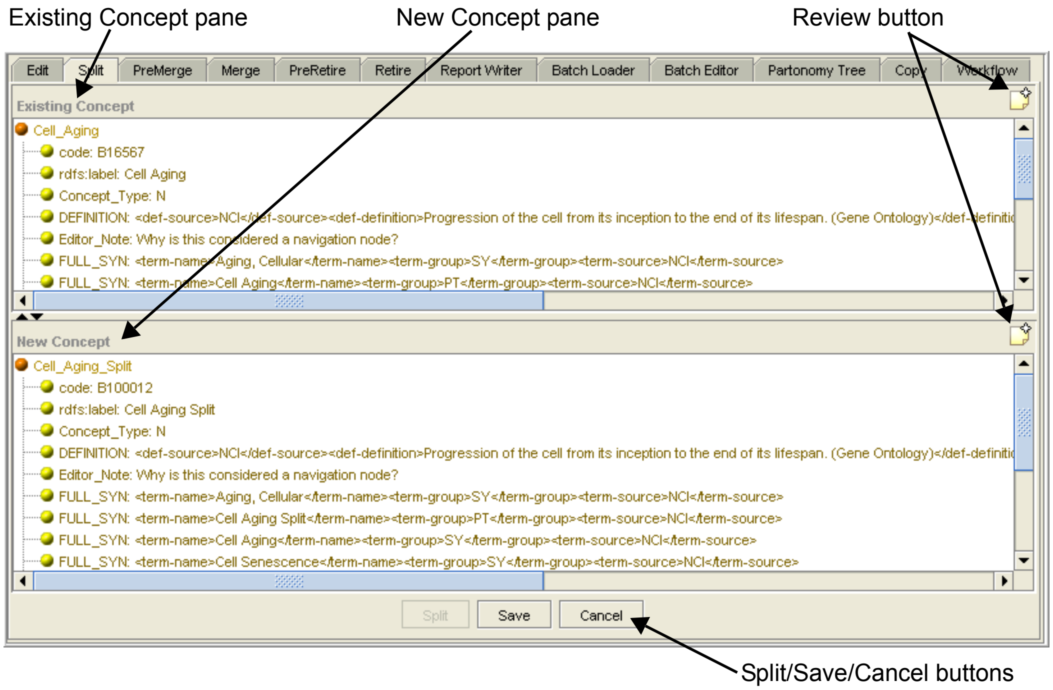
A class is typically *split* when you determine that not all of the atoms have the same meaning but seem to represent multiple classes. This could have been caused by user error, or it could indicate a new understanding of the meaning of the class.

The process of splitting a class creates a new class and moves the appropriate atoms to the new class. The new class becomes the sibling of the original class, and it automatically inherits all of the subclasses of the original class. To establish an audit trail, the new class is assigned a *Split_From* annotation property with a value equal to the code of the original class.

Note: Splitting a class is different from cases where atoms in one class actually belong to a different class. In such a case, simply remove those atoms and add them to the correct class.

About the Split Subtab

The Split subtab enables you to create a new class from an existing class. This subtab includes upper and lower horizontal panes for, respectively, existing and new concepts (*Figure 8.1*).



```
Cell_Aging
    code: B16567
    rdfs:label: Cell Aging
    Concept_Type: N
    DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontology)</def-definiti
    Editor_Note: Why is this considered a navigation node?
    FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Aging</term-name><term-group>PT</term-group><term-source>NCI</term-source>
```

```
Cell_Aging_Split
    code: B100012
    rdfs:label: Cell Aging Split
    Concept_Type: N
    DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontology)</def-definiti
    Editor_Note: Why is this considered a navigation node?
    FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Aging Split</term-name><term-group>PT</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Aging</term-name><term-group>SY</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Senescence</term-name><term-group>SY</term-group><term-source>NCI</term-source>
```

Figure 8.1 Split subtab

Table 8.1 describes the features of the Split subtab.

Feature	***Description***
Existing Concept pane (upper)	Shows details for a selected class.
New Concept pane (lower)	Shows details for a new class that has been created from the existing class shown in the upper pane.
Review button	Opens a Review window that shows corresponding class in its *original* state.
Split button	Creates a new class based on the existing class and displays the new class in the lower pane.
Save button	Saves the newly created class.
Cancel button	Discards unsaved data.

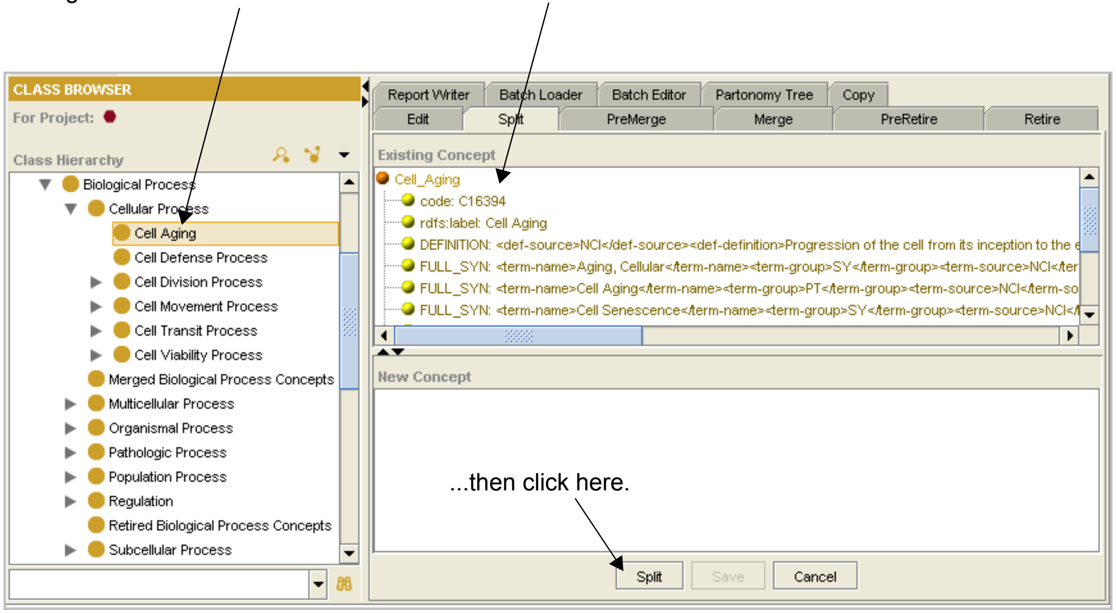
Table 8.1 Split subtab features

Using the Split Subtab to Split a Class

To split a class, follow these steps:

1. Using the Class Browser, select the class to be split.
2. Click the **Split** subtab.
3. Drag the selected class from the Class Browser into the **Existing Concept** pane (*Figure 8.2*).
4. Click the **Split** button at the bottom of the subtab.

Drag a class from here... ...to here...



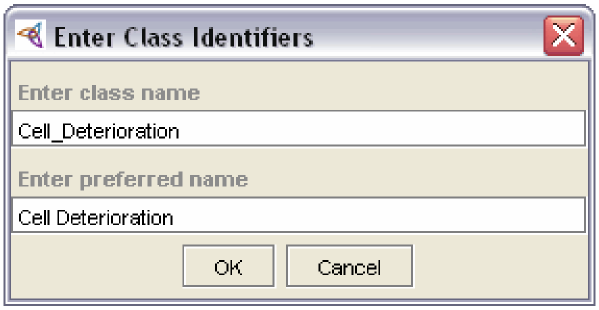
Cell_Aging
code: C16394
rdfs:label: Cell Aging
DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the e
FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</ter
FULL_SYN: <term-name>Cell Aging</term-name><term-group>PT</term-group><term-source>NCI</term-so
FULL_SYN: <term-name>Cell Senescence</term-name><term-group>SY</term-group><term-source>NCI</

...then click here.

Figure 8.2 Existing Concept pane

The Enter Class Identifiers window opens.

5. Enter the new class name with underscores and the preferred name without underscores (*Figure 8.3*).

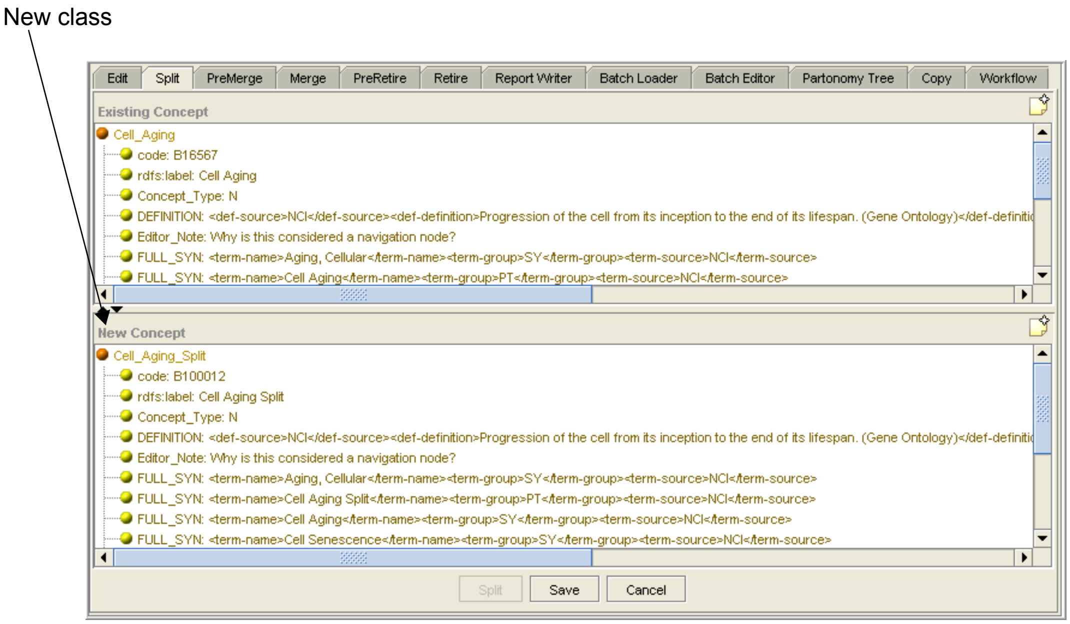


Enter class name
Cell_Deterioration
Enter preferred name
Cell Deterioration

Figure 8.3 Enter Class Identifiers window

6. Click **OK**.

The new class appears in the lower pane on the Split subtab (*Figure 8.4*).



```
Cell_Aging
    code: B16567
    rdfs:label: Cell Aging
    Concept_Type: N
    DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontology)</def-definiti
    Editor_Note: Why is this considered a navigation node?
    FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Aging</term-name><term-group>PT</term-group><term-source>NCI</term-source>
```

```
Cell_Aging_Split
    code: B100012
    rdfs:label: Cell Aging Split
    Concept_Type: N
    DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontology)</def-definiti
    Editor_Note: Why is this considered a navigation node?
    FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Aging Split</term-name><term-group>PT</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Aging</term-name><term-group>SY</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Senescence</term-name><term-group>SY</term-group><term-source>NCI</term-source>
```

Figure 8.4 Split classes

7. Use the right-click menu to add or modify properties:
    - Right-click the **class name** to do any of the following:
        - — Add a property
        - — Create a restriction
        - — Add an association
        - — Add a parent
        - — Create a workflow task (workflow managers only)
    - Right-click a **property** to do any of the following:
        - — Delete the property
        - — Modify the property
        - — Create a workflow task (workflow managers only)
        - — Copy the property
        - — Paste a copied property
8. Click the **Save** button to save the changes.

    A message confirms that the class has been saved successfully.
9. Click **OK** to close the message window.

Examining the Properties of a Split Class

Newly split classes appear in the Class Hierarchy. To examine the properties of a newly split class, follow these steps:

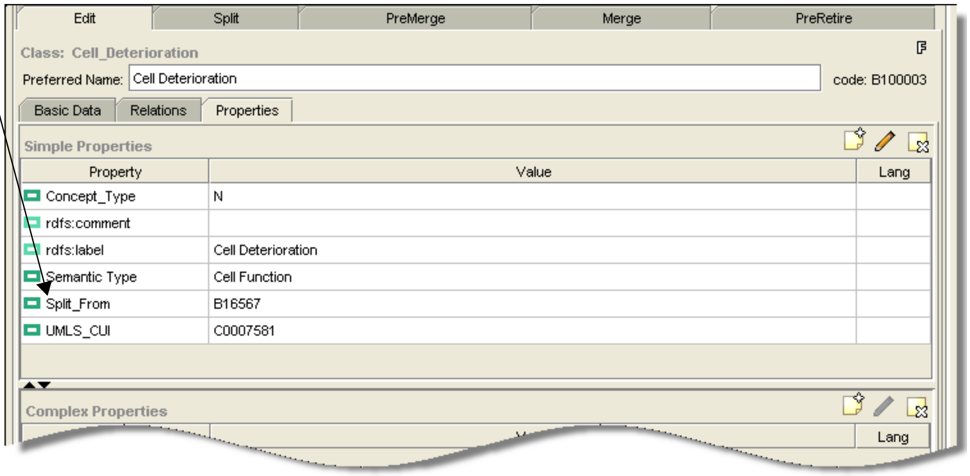
1. Locate the new class in the Class Hierarchy.

   The new class appears as a sibling of the existing class.

2. Click the **Edit** subtab to view the properties and restrictions for the new class.

3. On the **Properties** subtab, note the *Split_From* property, which shows the code of the existing concept.

Split_From property



Class: Cell_Deterioration

Preferred Name: Cell Deterioration

code: B100003

Simple Properties

Property	Value	Lang
Concept_Type	N	
rdfs:comment		
rdfs:label	Cell Deterioration	
Semantic Type	Cell Function	
Split_From	B16567	
UMLS_CUI	C0007581	

Complex Properties

Lang

Figure 8.5 Properties of new class

Merging Classes

The following topics explain how to flag classes for a merge (pre-merge) and how to merge classes (workflow managers only):

- *When to Consider Merging Classes* on this page
- *About the PreMerge Subtab* on page 125
- *Using the PreMerge Subtab to Flag Classes for a Merge* on page 126
- *Finding the Retiring Class under the Premerged_Concepts Branch* on page 128
- *About the Merge Subtab* on page 128
- *Using the Merge Subtab to Merge Flagged Classes* on page 129

When to Consider Merging Classes

Classes are typically *merged* when you determine that they are *synonymous*—that their meanings are not distinct from each other or are identical to an existing class.

Unless otherwise specified, the older class (the one with the lower code) survives and gains all of the roles, properties, parents, and children of the newer class. The newer class is retired. You can override this convention in either of the following cases:

- The newer class is more frequently referenced; or
- The newer class is better formed or more fully modeled.

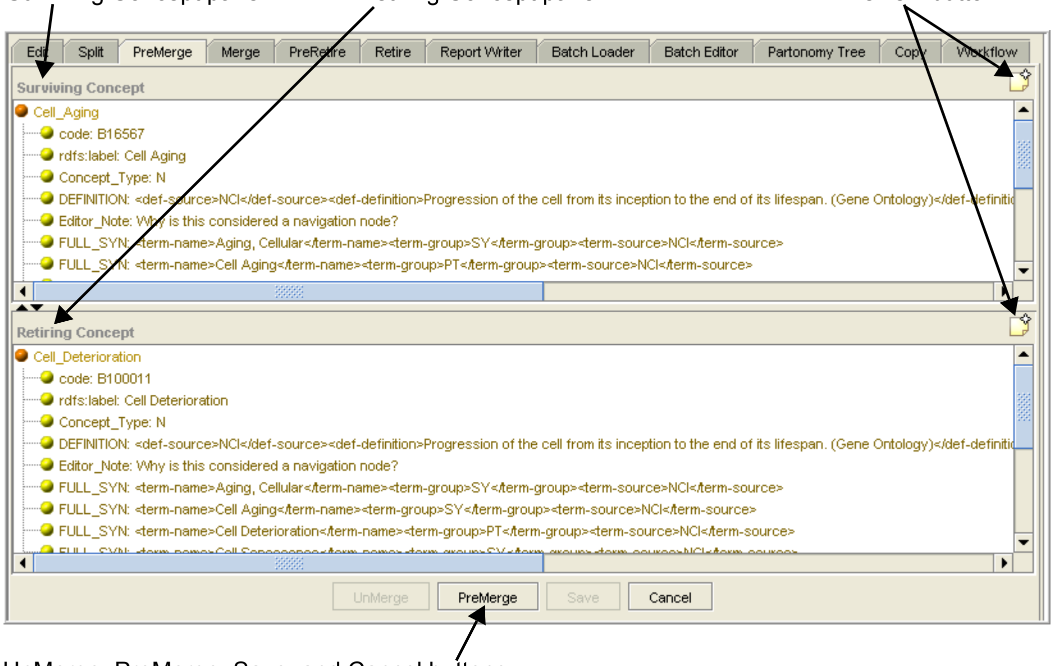
Merging is a two-step process:

1. An *editor* (modeler) applies a *pre-merge* flag to designate two classes for a merge. The retiring class is moved to a Premerged_Concepts branch in the class hierarchy.
2. During a baseline review and update, the *workflow manager* reviews the merge candidates and either accepts or rejects them.

Note: A merge is not a solution for changing the name of a class that is otherwise adequate. Instead, change the preferred name of the class.

About the PreMerge Subtab

The PreMerge subtab enables you to flag two classes for a merge. *Figure 8.6* shows the layout of this subtab.



Cell_Aging
- code: B16567
- rdfs:label: Cell Aging
- Concept_Type: N
- DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontology)</def-definiti
- Editor_Note: Why is this considered a navigation node?
- FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cell Aging</term-name><term-group>PT</term-group><term-source>NCI</term-source>

Cell_Deterioration
- code: B100011
- rdfs:label: Cell Deterioration
- Concept_Type: N
- DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontology)</def-definiti
- Editor_Note: Why is this considered a navigation node?
- FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cell Aging</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cell Deterioration</term-name><term-group>PT</term-group><term-source>NCI</term-source>

Figure 8.6 PreMerge subtab

Table 8.2 lists the features of the PreMerge subtab.

Feature	***Description***
Surviving Concept pane	Shows details for the *surviving* concept.
Retiring Concept pane	Shows details for the *retiring* concept.
Review button	Opens a Review window that shows the class in its *original* state.
UnMerge button	Removes a previously set pre-merge flag.
PreMerge button	Flags two classes for a merge.
Save button	Saves classes with the following annotation properties: • Surviving = *Merge_Source* • Retiring = *Merge_Target* **Note:** The value of each property is the code of the referenced class.
Cancel button	Reverses a pre-merge operation.

Table 8.2 PreMerge subtab features

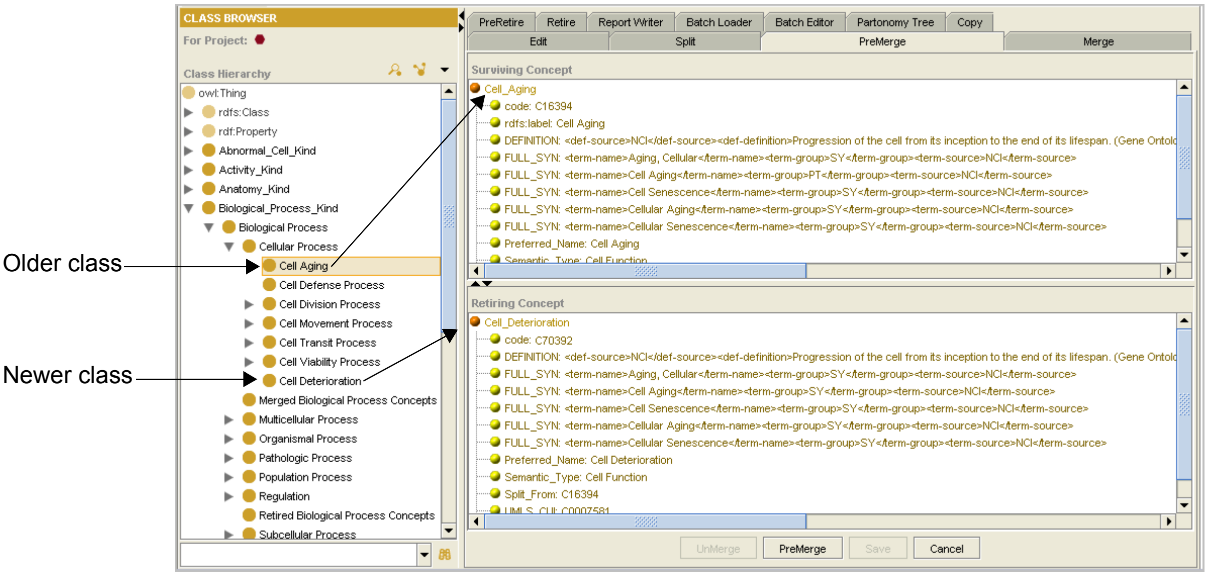
Using the PreMerge Subtab to Flag Classes for a Merge

To flag two classes for a merge, follow these steps:

1. Using the Class Browser, select one of the two classes to be flagged for a pre-merge.
2. Click the **PreMerge** subtab.
3. In the Class Browser, follow these steps:
    a. Select the class with the *higher* code (the newer class), then drag it into the **Retiring Concept** pane—the lower pane on the PreMerge subtab.

    b. Select the class with the *lower* code (the older class), then drag it into the **Surviving Concept** pane—the upper pane on the PreMerge subtab.

    Figure 8.7 illustrates the paths from the Class Browser to the Retiring Concept and Surviving Concept panes.



Surviving Concept

Cell_Aging
- code: C16394
- rdfs:label: Cell Aging
- DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontolo
- FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cell Aging</term-name><term-group>PT</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cell Senescence</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cellular Aging</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cellular Senescence</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- Preferred_Name: Cell Aging
- Semantic_Type: Cell Function

Retiring Concept

Cell_Deterioration
- code: C70392
- DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontolo
- FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cell Aging</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cell Senescence</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cellular Aging</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- FULL_SYN: <term-name>Cellular Senescence</term-name><term-group>SY</term-group><term-source>NCI</term-source>
- Preferred_Name: Cell Deterioration
- Semantic_Type: Cell Function
- Split_From: C16394
- UMLS_CUI: C0007581

Figure 8.7 PreMerge subtab with surviving and retiring concepts

Note: If you drag the older class to the lower pane and the newer class to the upper pane, Protégé reminds you that the retiring concept is older than the surviving concept and asks if you want to swap the two classes. If you click **Yes**, the classes are swapped.

4. Use the right-click menu to add or modify properties:
    - Right-click the **class name** to do any of the following:
        - Add a property
        - Create a restriction
        - Add an association
        - Add a parent
        - Create a workflow task (workflow managers only)

- Right-click a **property** to do any of the following:
  - Delete the property
  - Modify the property
  - Create a workflow task (workflow managers only)
  - Copy the property
  - Paste a copied property

5. Click the **Save** button to save the changes.

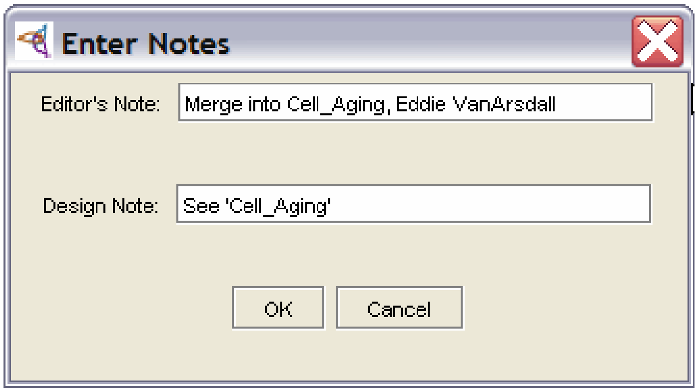
   A message confirms that the class has been saved successfully.

6. Click **OK** to close the message window.
7. When you are ready to complete the pre-merge, click the **PreMerge** button at the bottom of the subtab.

   The Enter Notes window opens and displays the following pre-filled information (*Figure 8.8*):

   - The Editor's Note is for internal use only. Since this note is not published, you can use it to provide information and directions for other editors.
   - The Design Note is for all Thesaurus users. Use this note to provide additional information about the meaning and use of the concept.

8. Modify the Editor's Note and Design Note if necessary.



Editor's Note: Merge into Cell_Aging, Eddie VanArsdall

Design Note: See 'Cell_Aging'

Figure 8.8 Enter Notes window

9. Click **OK** to close the Enter Notes window.

   The PreMerge button is now disabled, and the UnMerge button is available. To undo the PreMerge flag, click the **UnMerge** button.

10. Click the **Save** button.

    A message confirms that the two classes have been flagged for merge.

11. Click **OK** to close the message window.

    The Surviving Concept and Retiring Concept panes are now empty.

Finding the Retiring Class under the Premerged_Concepts Branch

To locate a retiring class that you have flagged for a merge, follow these steps:

1. Locate the **Premerged_Concepts** branch in the Class Hierarchy.

   The Premerged_Concepts branch displays classes that are flagged for a merge.

2. Expand the branch and select the class.
3. Examine the class properties under the **Edit** subtab > **Properties** subtab.

   The Simple Properties panel now shows a *Merge_Target* property for the retiring class.

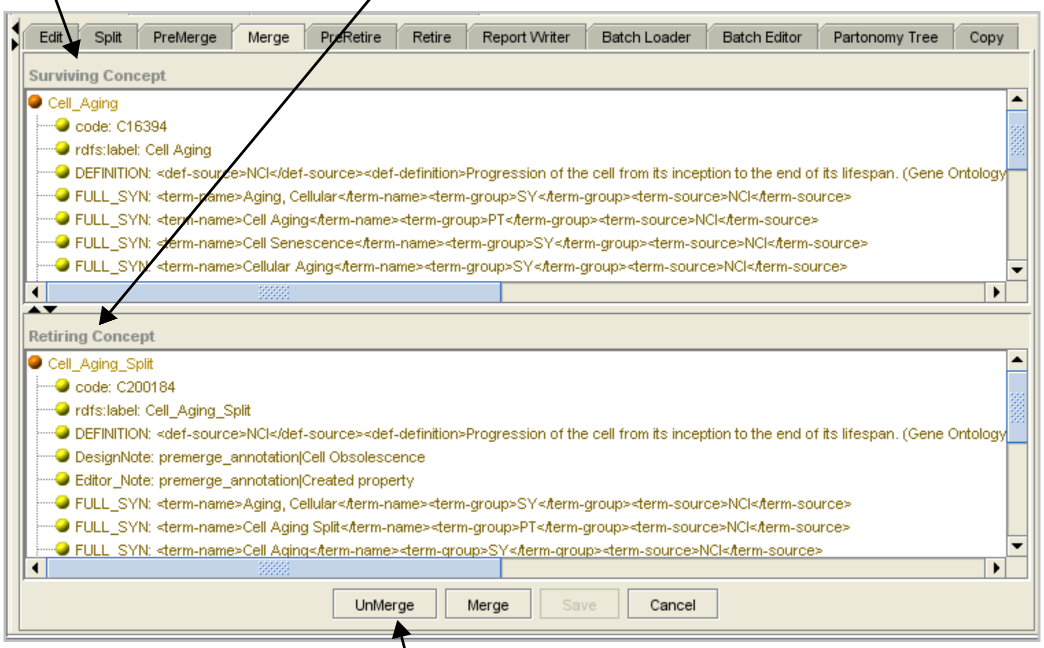
About the Merge Subtab

The Merge subtab enables workflow managers to merge two classes that have been flagged using the pre-merge action. For more information about the *pre-merge* action, see *Using the PreMerge Subtab to Flag Classes for a Merge* on page 126.

Figure 8.9 shows the layout of the Merge subtab.

Surviving Concept pane

Retiring Concept pane



```
Cell_Aging
    code: C16394
    rdfs:label: Cell Aging
    DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontology
    FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Aging</term-name><term-group>PT</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Senescence</term-name><term-group>SY</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cellular Aging</term-name><term-group>SY</term-group><term-source>NCI</term-source>
```

```
Cell_Aging_Split
    code: C200184
    rdfs:label: Cell_Aging_Split
    DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception to the end of its lifespan. (Gene Ontology
    DesignNote: premerge_annotation|Cell Obsolescence
    Editor_Note: premerge_annotation|Created property
    FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Aging Split</term-name><term-group>PT</term-group><term-source>NCI</term-source>
    FULL_SYN: <term-name>Cell Aging</term-name><term-group>SY</term-group><term-source>NCI</term-source>
```

UnMerge, Merge, Save, and Cancel buttons

Figure 8.9 Merge subtab

Table 8.3 lists the features of the Merge subtab.

Feature	***Description***
Surviving Concept pane	Shows a tree representation of a selected concept that has been flagged for a merge. This concept has a *Merge_Source* annotation property.
Retiring Concept pane	Shows a tree representation of a selected concept that has been designated as a retiring class.
UnMerge button	Removes a previously set pre-merge flag.
Merge button	Flags two classes for a merge.
Save button	Merges two classes and saves the change.
Cancel button	Reverses a pre-merge operation.

Table 8.3 Merge subtab features

Using the Merge Subtab to Merge Flagged Classes

Note: Only workflow managers can perform this procedure.

To merge flagged classes, follow these steps:

1. Using the Class Browser, locate and expand the **Premerged_Concepts** branch.
2. Locate and select the class that will be retired as a result of the merge.
3. Click the **Merge** subtab.
4. Drag the selected class into the **Retiring Concept** (lower) pane on the right.

   The Surviving Concept and Retiring Concept panes now show, respectively, the two classes used in the PreMerge action.
5. Click the **Merge** button at the bottom of the Merge subtab.

   The Merge button is now disabled, and the UnMerge button is available. If you want to undo the merge, click the **UnMerge** button.
6. Click the **Save** button to accept the change.

   A message confirms that the two classes have been merged successfully.
7. Click **OK** to close the message window.

   The two panes on the Merge subtab are now empty. The retired class has been removed from the Premerged_Concepts branch of the Class Hierarchy.

Cloning a Class

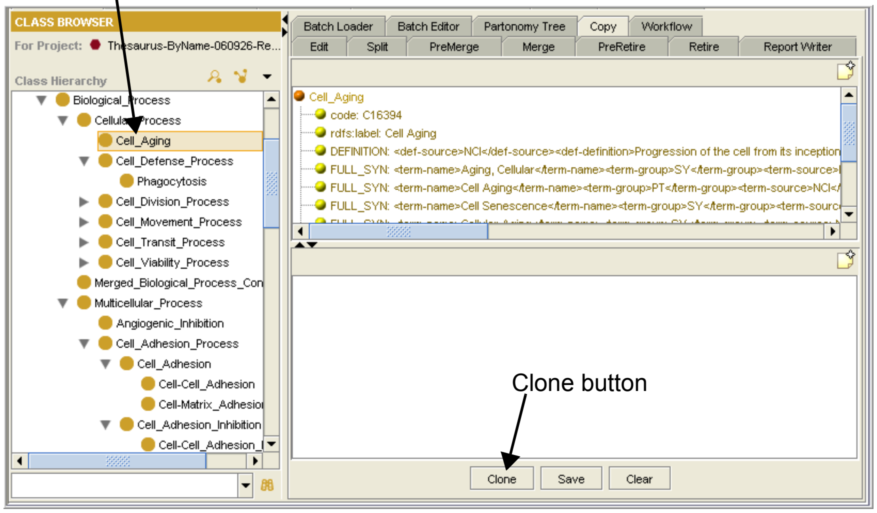
Protégé enables you to clone a new class from an existing class, with the original class essentially serving as a template for the new class.

To clone a class, follow these steps:

1. In the main Protégé window, click the **Copy** subtab.
2. Using the Class Browser, drag a class into the upper pane (*Figure 8.10*).

   If you drag a class into the lower pane and the upper pane is empty, Protégé automatically places the class in the upper pane.
3. Click the **Clone** button.

Original class



Cell_Aging
code: C16394
rdfs:label: Cell Aging
DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception
FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>
FULL_SYN: <term-name>Cell Aging</term-name><term-group>PT</term-group><term-source>NCI<
FULL_SYN: <term-name>Cell Senescence</term-name><term-group>SY</term-group><term-sourc

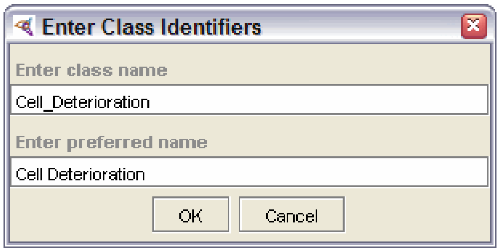
Clone button

Figure 8.10 Copy subtab with original class

The Enter Class Identifiers window opens.

4. Enter the class name and preferred name (*Figure 8.11*).

   Note: Remember that the class name requires underscores between words, whereas the preferred name can include spaces.

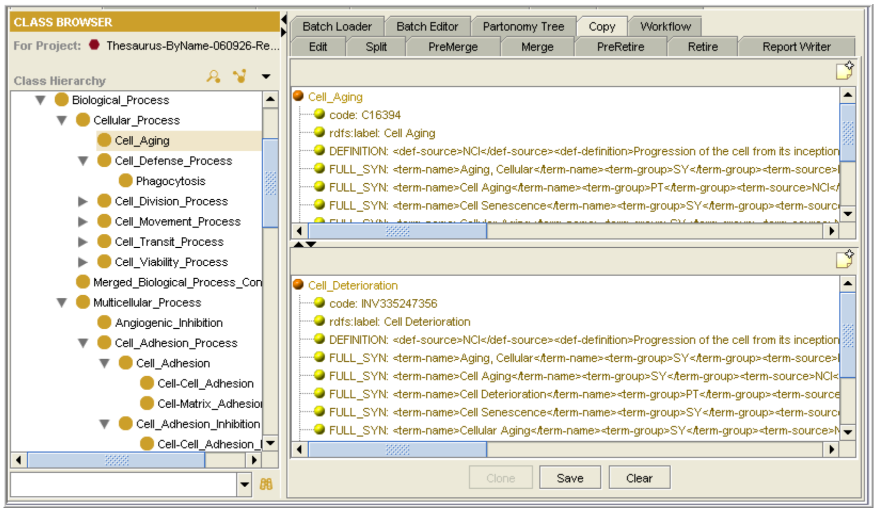


Enter class name
Cell_Deterioration
Enter preferred name
Cell Deterioration

Figure 8.11 Enter Class Identifiers window

5. Click **OK** to close the Enter Class Identifiers window.

The new class now appears in the lower pane of the Copy subtab (*Figure 8.12*).



Cell_Aging
code: C16394
rdfs:label: Cell Aging
DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception
FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>
FULL_SYN: <term-name>Cell Aging</term-name><term-group>PT</term-group><term-source>NCI</
FULL_SYN: <term-name>Cell Senescence</term-name><term-group>SY</term-group><term-sourc

Cell_Deterioration
code: INV335247356
rdfs:label: Cell Deterioration
DEFINITION: <def-source>NCI</def-source><def-definition>Progression of the cell from its inception
FULL_SYN: <term-name>Aging, Cellular</term-name><term-group>SY</term-group><term-source>
FULL_SYN: <term-name>Cell Aging</term-name><term-group>SY</term-group><term-source>NCI<
FULL_SYN: <term-name>Cell Deterioration</term-name><term-group>PT</term-group><term-source
FULL_SYN: <term-name>Cell Senescence</term-name><term-group>SY</term-group><term-sourc
FULL_SYN: <term-name>Cellular Aging</term-name><term-group>SY</term-group><term-source>N

Figure 8.12 Copy subtab with newly created class

Tip: To start over without saving the newly cloned class, click the **Clear** button below the Copy subtab. A prompt ensures that you do not want to save the new class, then clears the top and bottom panes. You can also use the **Clear** button to clear the panes *after saving* a newly cloned class.

6. Use the right-click menu to add or modify properties:
   - Right-click the **class name** to do any of the following:
     - — Add a property
     - — Create a restriction
     - — Add an association
     - — Add a parent
     - — Create a workflow task (workflow managers only)
   - Right-click a **property** to do any of the following:
     - — Delete the property
     - — Modify the property
     - — Create a workflow task (workflow managers only)
     - — Copy the property
     - — Paste a copied property
7. (Optional) Click the **Review** button 📝 in the upper or lower pane to view the state of either class before saving.
8. Click the **Save** button to save the changes.
9. When the confirmation message appears, click **OK** to close the window.

CHAPTER

9

PRE-RETIRING AND RETIRING CLASSES

This chapter explains the processes and procedures for pre-retiring and retiring a class.

Topic in this Chapter

About the Retirement Process

This topic explains when you might want to retire a class. It also explains the roles of the editor and the workflow manager in the retirement process.

When to Consider Retiring a Class

A class is typically retired when an editor determines that it is no longer needed. For example, another better-modeled class may have the same meaning.

PreRetire vs. Retire

Retiring is a two-step process:

1. An *editor* (modeler) applies a *PreRetire* flag, which designates a class for retirement.

   The flagged class is moved to a **Preretired_Concepts** branch in the Class Hierarchy.

2. During a baseline review and update, the *workflow manager* reviews the pre-retired candidates and either accepts or rejects them.

   Retired classes are deprecated but remain accessible in search results. Their names are preceded by a letter D enclosed in square brackets: [D].

Pre-Retiring a Class

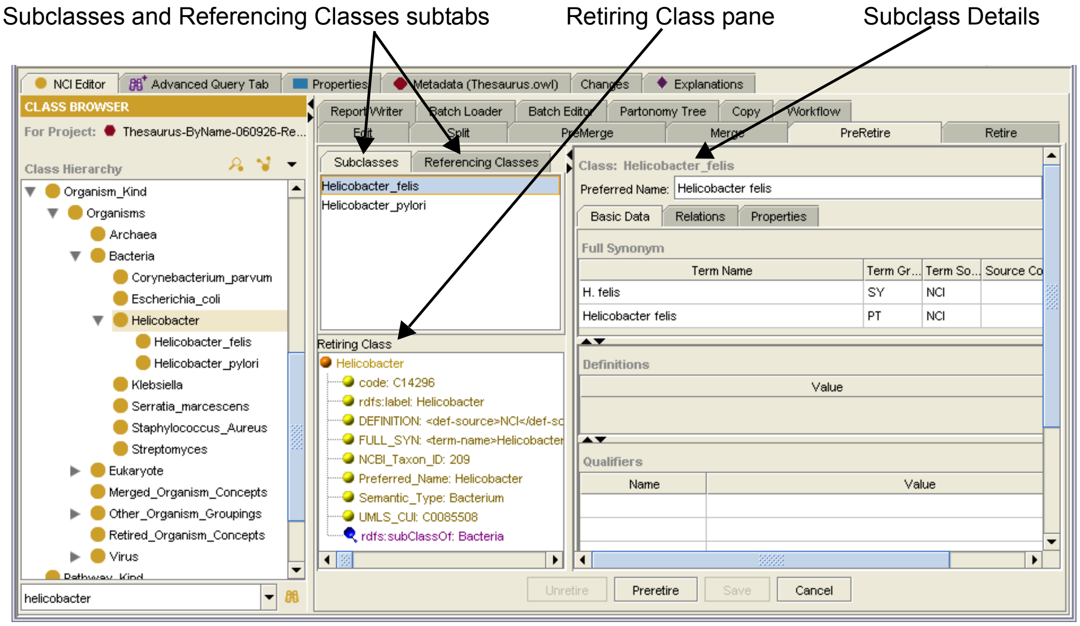
The following topics introduce the PreRetire subtab, explain how to use it to add a pre-retire flag to a class, and explain where to locate pre-retired classes:

- *About the PreRetire Subtab* on this page
- *Using the PreRetire Subtab to Flag a Class for Retirement* on page 135
- *Finding a Pre-Retired Class in the Hierarchy* on page 139

About the PreRetire Subtab

The PreRetire subtab enables you to flag a class for retirement. This subtab includes two subtabs that show, respectively, subclasses and referencing classes.

Figure 9.1 shows the layout of the PreRetire subtab.


Subclasses and Referencing Classes subtabs

Retiring Class pane

Subclass Details

Figure 9.1 PreRetire subtab

Table 9.1 describes the features of the PreRetire subtab.

Feature	***Description***
Subclasses subtab	Shows the names of all subclasses of the currently selected class.
Referencing Classes subtab	Shows the names of all subclasses that have the currently selected class as a referred class.
Basic Data, Relations, and Properties subtabs	Show basic data, relations, and properties for a selected subclass or referencing class.
Retiring Class pane	Shows a tree representation of the selected class.
Unretire button	Reverses a pre-retire action.

Table 9.1 PreRetire subtab features

Feature	**Description**
Preretire button	Flags a selected class for retirement.
Save button	Saves a selected class with a pre-retire flag.
Cancel button	Discards unsaved changes.

Table 9.1 PreRetire subtab features

Using the PreRetire Subtab to Flag a Class for Retirement

You can designate a class for retirement by applying a pre-retire flag. To do this, perform the following procedures in the following order:

- *Selecting a Class for Retirement* on this page
- *Eliminating Subclass Dependencies* on page 135
- *Saving the Class with a Pre-Retired Status* on page 138

Selecting a Class for Retirement

To select a class for retirement, follow these steps:

1. In the main Protégé window, click the **PreRetire** subtab.
2. Using the Class Browser, drag a class into the Retiring Class panel on the lower left side of the PreRetire subtab (*Figure 9.1*).

   If subclasses exist for the selected class, they appear in the Subclasses panel on the upper left. If other classes reference the selected class by role relations, you can see them by clicking the Referencing Classes subtab.
3. Eliminate any subclass dependencies.

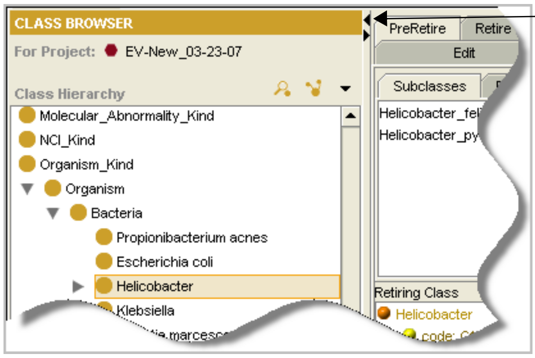
Eliminating Subclass Dependencies

When flagging a class for retirement, you first eliminate any dependencies for the class by assigning each of its subclasses to another parent class.

Prerequisite procedure: *Selecting a Class for Retirement* on page 135

To eliminate subclass dependencies, follow these steps for each subclass:

1. Click the **Expand** button on the top right to expand the **PreRetire** subtab and hide the Class Browser (*Figure 9.2*).

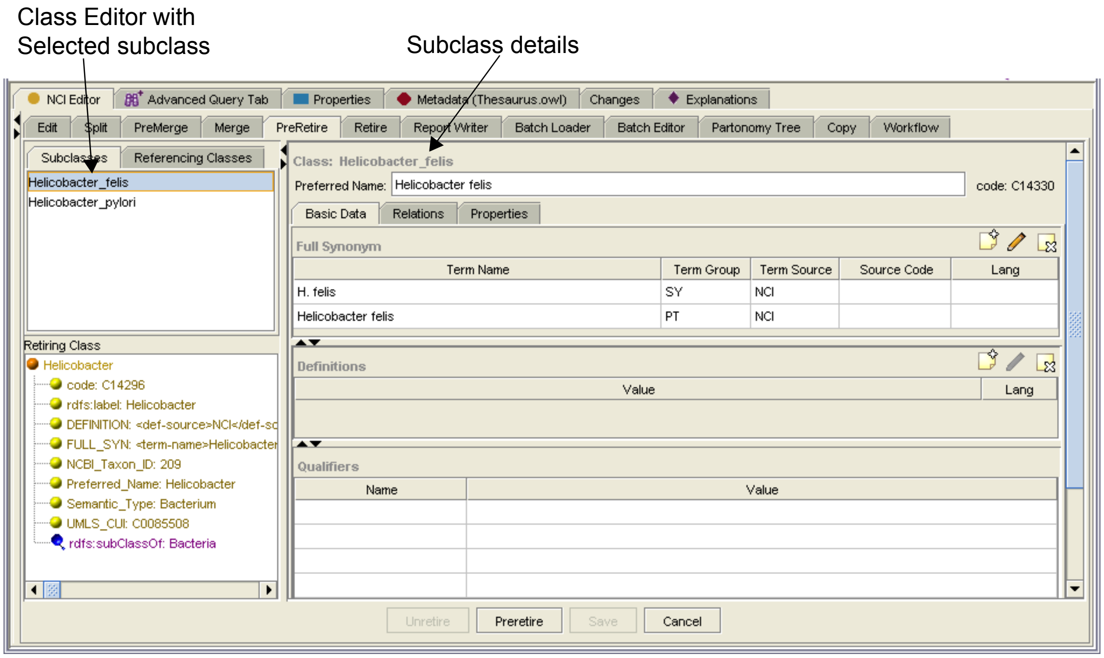


Expand button

Figure 9.2 Hiding the Class Browser

2. Select a subclass in the **Subclasses** panel (upper left).

   Detailed information for the selected subclass now appears in the Class Editor pane on the right (*Figure 9.3*).



Subclasses:
Helicobacter_felis
Helicobacter_pylori

Retiring Class
Helicobacter
code: C14296
rdfs:label: Helicobacter
DEFINITION: <def-source>NCI</def-sc
FULL_SYN: <term-name>Helicobacter
NCBI_Taxon_ID: 209
Preferred_Name: Helicobacter
Semantic_Type: Bacterium
UMLS_CUI: C0085508
rdfs:subClassOf: Bacteria

Class: Helicobacter_felis
Preferred Name: Helicobacter felis code: C14330

Full Synonym

Term Name	Term Group	Term Source	Source Code	Lang
H. felis	SY	NCI		
Helicobacter felis	PT	NCI		

Definitions

Value	Lang

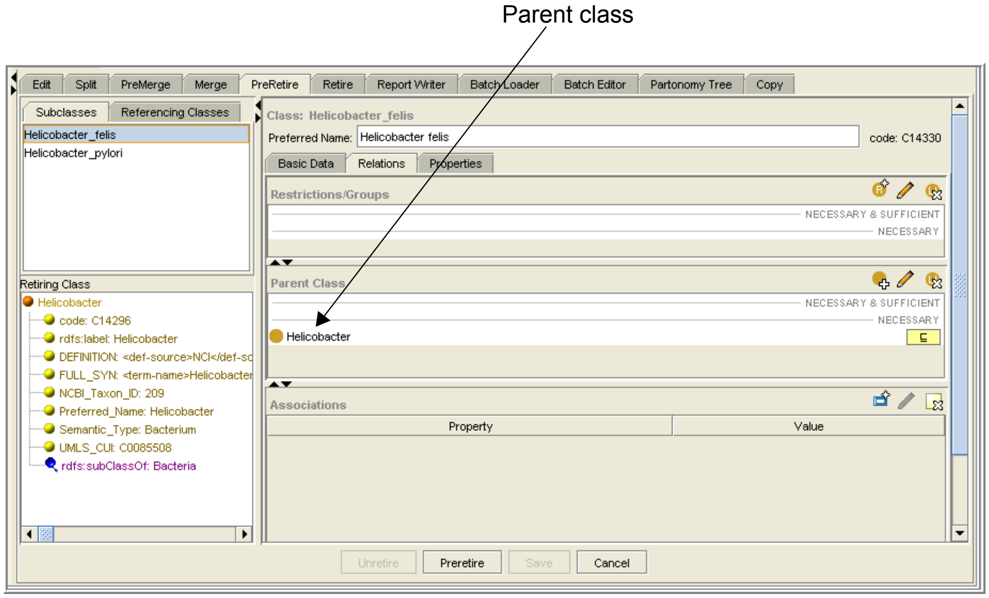
Qualifiers

Name	Value

Figure 9.3 Selected subclass with Basic Data subtab displayed

3. Click the **Relations** subtab to view parent classes for the selected class.

   The current example has only one parent class: *Helicobacter* (*Figure 9.4*). When only one named parent class exists, you cannot delete it.



Subclasses:
Helicobacter_felis
Helicobacter_pylori

Retiring Class
Helicobacter
code: C14296
rdfs:label: Helicobacter
DEFINITION: <def-source>NCI</def-sc
FULL_SYN: <term-name>Helicobacter
NCBI_Taxon_ID: 209
Preferred_Name: Helicobacter
Semantic_Type: Bacterium
UMLS_CUI: C0085508
rdfs:subClassOf: Bacteria

Class: Helicobacter_felis
Preferred Name: Helicobacter felis code: C14330

Parent Class: Helicobacter

Associations

Property	Value

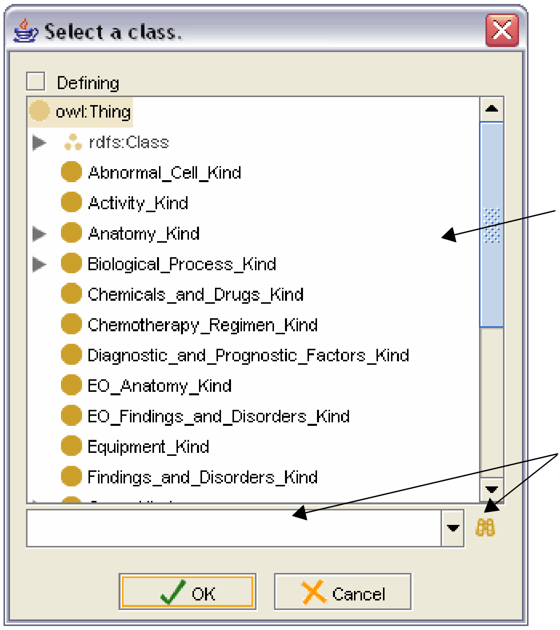
Figure 9.4 Relations subtab with one parent class

To retree the subclass *Helicobacter_felis*, you first need to add a second superclass, as in the following steps.

4. Click the **Add parent class** button in the upper right area of the Parent Class panel.

   The Select a class window opens (*Figure 9.5*).

5. Select or search for a class.



Select a class...

...or search for a class using the search field and button.

Figure 9.5 Select a class window

Note: If you search for a class, the Advanced Query window opens. Select a search result, then click **OK** to close the window and return to the Select a superclass window.

6. Click **OK** to close the Select a class window.

   In the current example, there are now two parent classes: *Bacteria* and *Helicobacter*.

7. Select the class to be deleted.

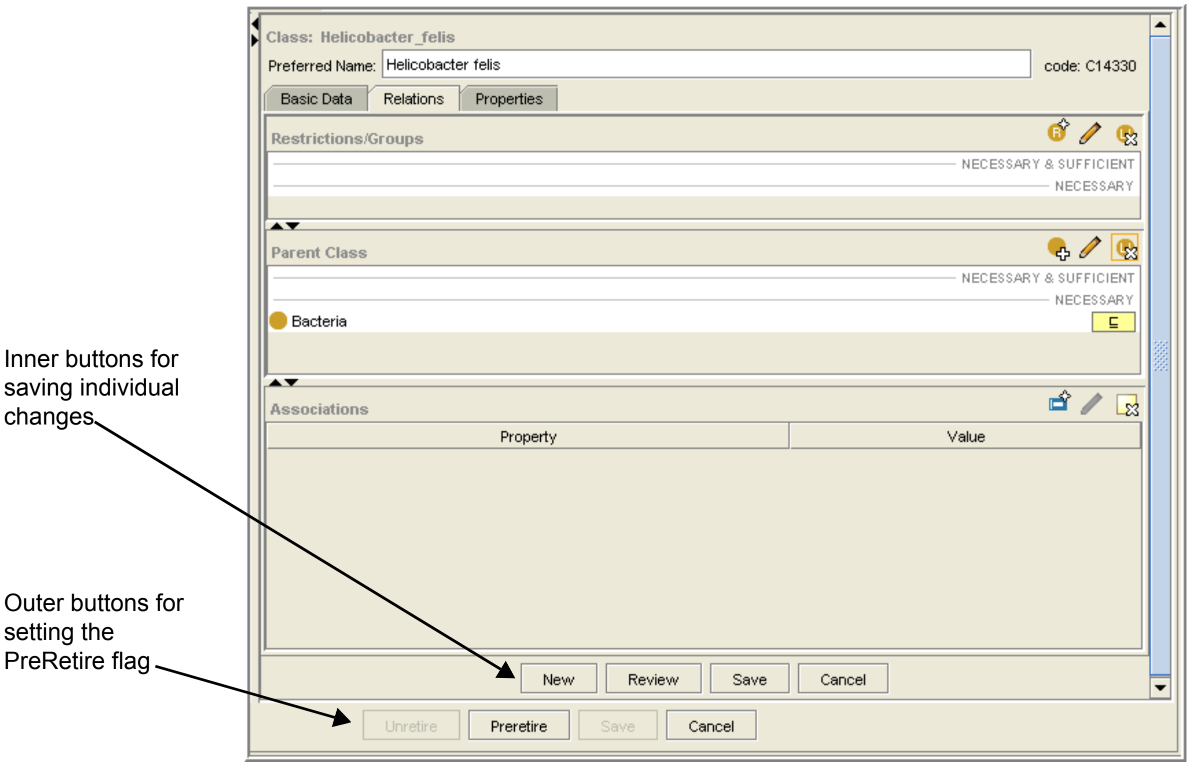
   For the current example, select the **Helicobacter** class.

8. Click the **Delete selected row** button in the upper right area of the Parent Class panel.

9. Click **Yes** to confirm deletion of the selected class.

Note: The Basic Data, Relations, and Properties subtabs are nested inside of the PreRetire subtab, so you may not notice that there are two separate sets of buttons at the bottom (*Figure 9.6*).

In the next step, make sure that you click the **Save** button at the bottom of the *inner* subtab to re-tree each subclass. When you are ready to flag the class for retirement, click the **PreRetire** button at the bottom of the *outer* subtab. You will then be asked to save the change using the *outer* **Save** button.



Class: Helicobacter_felis

Preferred Name: Helicobacter felis

code: C14330

Inner buttons for saving individual changes

Outer buttons for setting the PreRetire flag

Figure 9.6 Buttons sets for inner, nested subtabs and outer subtab

10. Click the *inner* **Save** button to accept the change.

    The selected subclass disappears from the Subclasses list.

11. Click **OK** to close the confirmation message window.

12. Repeat steps 2 through 11 for all classes shown in the Subclasses panel.

    When you are finished, there should be no subclasses shown.

13. Save the class with a Pre-Retired status.

Saving the Class with a Pre-Retired Status

Prerequisite procedure: *Eliminating Subclass Dependencies* on page 135

To pre-retire the class and save the change, follow these steps:

1. Click the **PreRetire** button at the bottom of the PreRetire subtab.

   The Enter Notes window opens and displays the following fields (*Figure 9.7*):

   - The Editor’s Note is for internal use only. Since this note is not published, you can use it to provide information and directions for other editors.
   - The Design Note is for all Thesaurus users. Use this note to provide additional information about the meaning and use of the concept.

2. Modify the Editor’s Note and Design Note if necessary.

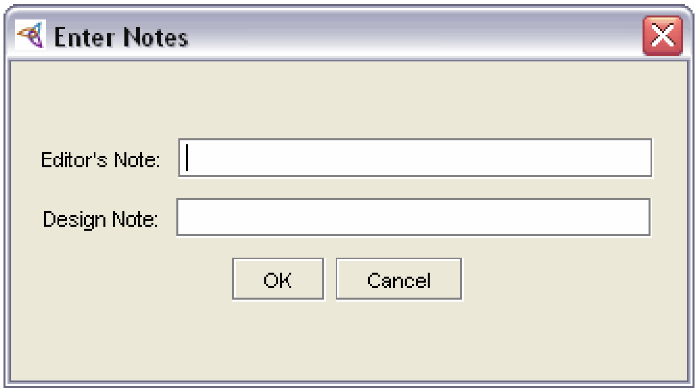


Figure 9.7 Enter Notes window

3. Click **OK** to close the Enter Notes window.
4. Click the *outer* **Save** button to accept the change.

   A message confirms that the selected class has been flagged for retirement.
5. Click **OK** to close the message window.

Finding a Pre-Retired Class in the Hierarchy

To locate the pre-retired class, follow these steps:

1. Locate the **Preretired_Concepts** branch in the Class Hierarchy.

   The Preretired_Concepts branch displays classes that are retirement.
2. Expand the branch and select the class.
3. Examine the class properties under the **Edit** subtab > **Properties** subtab.

   Among other properties, the Simple Properties panel shows the following information:

   - Your design note
   - Your editor’s note
   - The re-treed subclasses and their parent

Retiring a Pre-Retired Class

During a baseline review and update, workflow managers review the candidates designated for retirement and either accept or reject them. The following procedures explain how to accept a class for retirement.

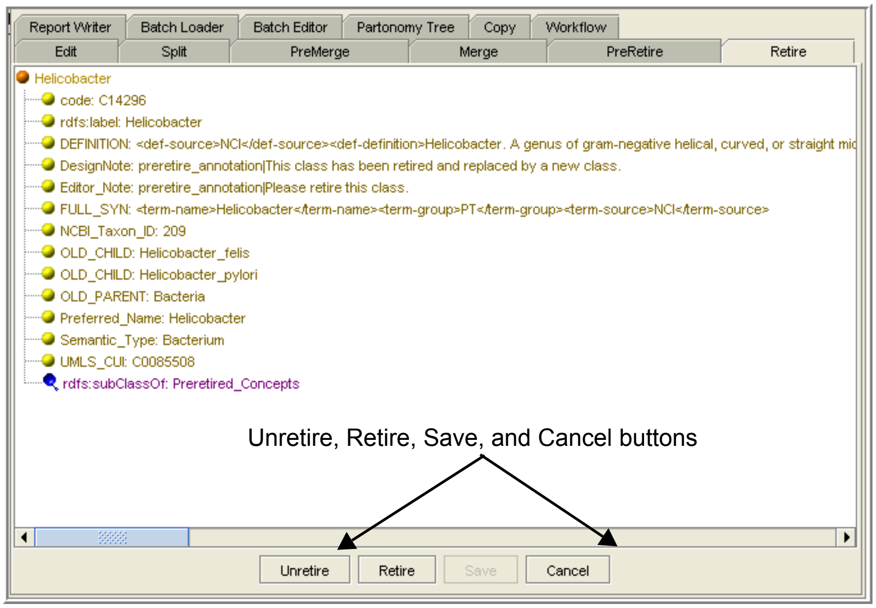
Note: Only workflow managers can retire a class.

- *About the Retire Subtab* on this page
- *Using the Retire Subtab to Retire a Flagged Class*

About the Retire Subtab

The Retire subtab enables authorized users (*e.g.*, workflow managers) to retire a class that has been flagged by a pre-retire action. This subtab has a single pane that shows a tree representation of the selected class.

Figure 9.8 shows the layout of the Retire subtab.



```
Helicobacter
  code: C14296
  rdfs:label: Helicobacter
  DEFINITION: <def-source>NCI</def-source><def-definition>Helicobacter. A genus of gram-negative helical, curved, or straight mi
  DesignNote: preretire_annotation|This class has been retired and replaced by a new class.
  Editor_Note: preretire_annotation|Please retire this class.
  FULL_SYN: <term-name>Helicobacter</term-name><term-group>PT</term-group><term-source>NCI</term-source>
  NCBI_Taxon_ID: 209
  OLD_CHILD: Helicobacter_felis
  OLD_CHILD: Helicobacter_pylori
  OLD_PARENT: Bacteria
  Preferred_Name: Helicobacter
  Semantic_Type: Bacterium
  UMLS_CUI: C0085508
  rdfs:subClassOf: Preretired_Concepts
```

Figure 9.8 Retire subtab

Table 9.2 lists the features of the Retire subtab.

Feature	***Description***
Unretire button	Removes a previously set pre-retire flag.
Retire button	Retires a class.
Save button	Saves changes and formally retires the class by re-treeing it from *pre-retired* to *retired*.
Cancel button	Discards unsaved changes.

Table 9.2 Retire subtab features

Using the Retire Subtab to Retire a Flagged Class

To retire a concept, follow these steps:

1. Using the Class Browser, locate the **Pretired_Concepts** branch.
2. Expand the branch, then select the class to be retired.
3. Click the **Retire** subtab.
4. Drag the selected class onto the **Retire** subtab.
5. Click the **Retire** button at the bottom of the subtab.
6. Click the **Save** button.

   A message confirms that the class has been retired.
7. Click **OK** to close the message window.

   The Retire subtab is now empty.

CHAPTER

10

USING NCI PROTÉGÉ UTILITIES

This chapter introduces three useful NCI Protégé utilities: the Report Writer, Batch Loader, and Batch Editor subtabs.

Topics in this Chapter

Generating Reports for Selected Classes

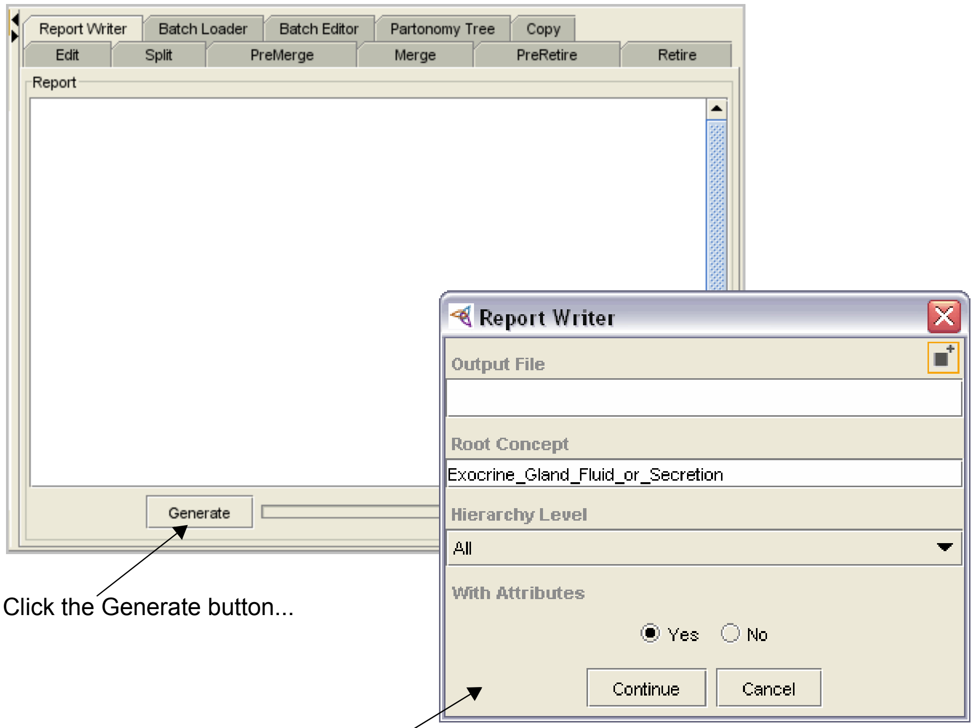
Protégé enables you to generate a report for a selected class. You can choose to have the report show only the parent class and its subclasses, or you can choose to also show the properties and restrictions for the class.

About the Report Writer Subtab

Use the Report Writer subtab to produce a report for a selected class. *Figure 10.1* (page 144) shows the simple Report Writer subtab interface, which is used mainly to open the Report Writer window.

In the window you can specify the name and location of an external report file (usually a text file). You can then view the report in a text editor such as Notepad.

Caution: When using the Report Writer, note that if you want to run a report on a high-level class in the tree (such as pharmacological agents), you may experience an extended wait time while the report runs. This could prevent you from performing other procedures in Protégé. The Report Writer will calculate the tree size before running the report and will enable you to choose whether to run the report or cancel the procedure.



Root Concept: Exocrine_Gland_Fluid_or_Secretion

Hierarchy Level: All

With Attributes: ● Yes ○ No

Click the Generate button...

...to open the Report Writer window

Figure 10.1 Report Writer subtab and Report Writer window

Using the Report Writer Subtab to Generate Reports

To generate reports, perform the following procedures in the following order:

- *Creating an Output Directory and Files* on this page
- *Generating a Report*

Creating an Output Directory and Files

Each report that you generate requires a specified output directory and file. For best results, create a local or network directory for storing report files.

Caution: If you do not specify a directory, Protégé stores report files in its own application directory (*C:\Program Files\Protege.Client-1.2.x*). If you store report files in the same directory as application files, you risk inadvertently deleting files that could cause Protégé not to run properly or not to run at all.

While naming report files, consider their purpose. For example, if you need a report for the class *Respiratory System Fluid or Secretion*, you can create one file that shows only the parent class and its children, and another file that shows parents, children, properties, and restrictions. Append the *.txt* extension to each file name, as in the following examples:

- *respiratory_system_parent-child-only.txt*
- *respiratory_system_all.txt*

Generating a Report

To generate a report for a selected class, follow these steps:

1. Using the Class Browser, select the class that will be the subject of the report.

   Note: This is a required step for generating a report. If you select the root of the hierarchy (*owl:Thing*), the Report Writer will prompt you to select another root.

2. Click the **Report Writer** subtab.

   As shown in *Figure 10.2*, the Report Writer subtab shows mostly white space. It does not display report output. Its main purpose is to generate an external text file that you can open and view in a text editor such as Notepad.

3. With the class still selected in the Class Browser on the left, click the **Generate** button (shown in *Figure 10.2*).

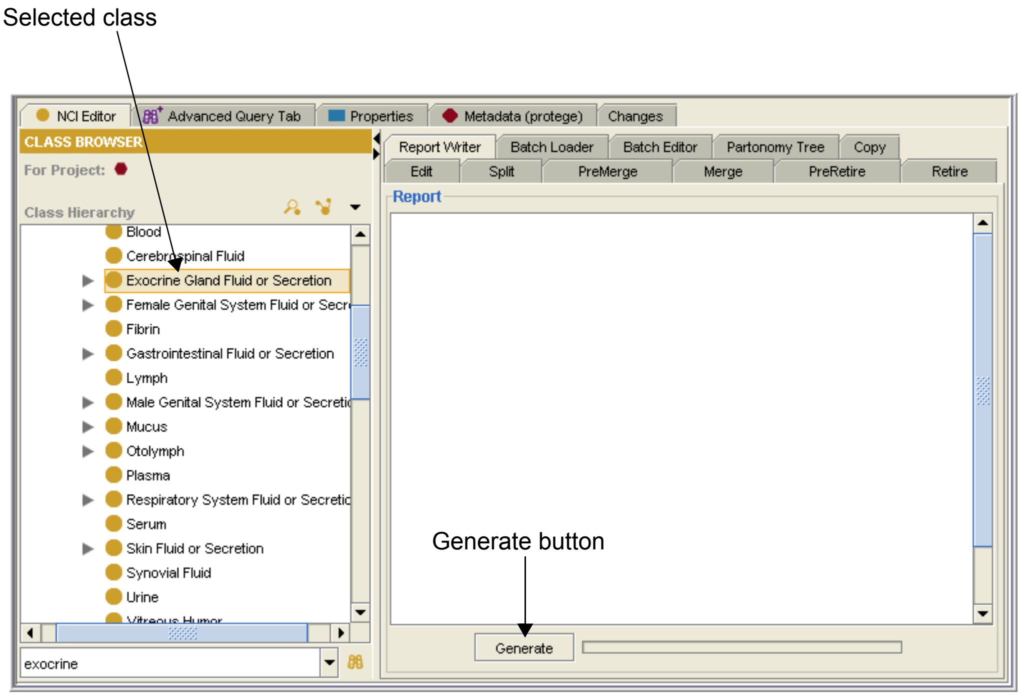


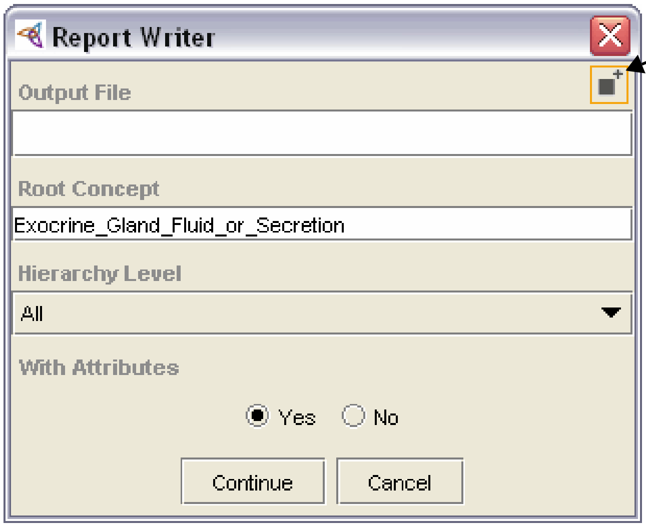
Figure 10.2 Report Writer subtab - Generate button

The Report Writer window opens. The selected class name is displayed in the **Root Concept** field (*Figure 10.3* on page 146).

Note: Although you can enter a new root concept name, the Report Writer will still generate a report for the class that you selected in the first step. Typing a new value will not change the result.

4. Select the output file for the report by following these steps:

   a. Click the **Browse for File** button in the upper right area of the window.


Browse for File button

Output File:

Root Concept: Exocrine_Gland_Fluid_or_Secretion

Hierarchy Level: All

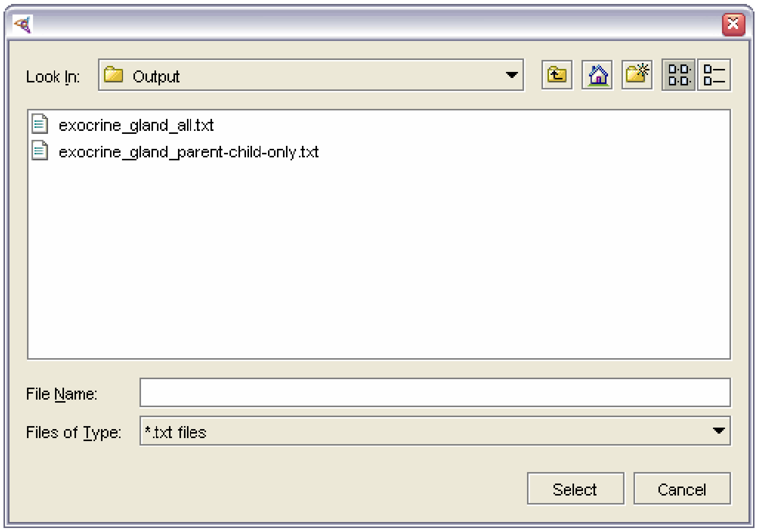
With Attributes: ● Yes ○ No

Figure 10.3 Report Writer window for specifying an output file

   b. In the File Browser window (shown in *Figure 10.4*), do either of the following:

      — If you are creating a *new* output file, browse to the location where the file should be stored, then type a name in the File Name field. Make sure that you include the `.txt` extension.

      — If you are using an existing output file, browse to the appropriate directory, then select the file.



Look In: Output

exocrine_gland_all.txt
exocrine_gland_parent-child-only.txt

File Name:

Files of Type: *.txt files

Figure 10.4 File browser window

   c. Click the **Select** button to close the file browser window.

Back in the Report Writer window, the file path and name now appear in the Output File field.

5. Specify the levels of the hierarchy to be included in the report by selecting a value from the hierarchy list.

   The **All** level and the first three numerical levels display the following information:

   - **All** includes the parent (root) class and all subclasses.
   - **Level zero (0)** displays only the parent class.
   - **Level one (1)** displays the parent class and its subclasses (children).
   - **Level two (2)** displays the parent class, its subclasses, and any children of the subclasses.

   The results of selecting the remaining levels vary according to the number of children that the selected parent has. *Figure 10.5* shows the level numbers of the current example, *Exocrine Gland Fluid or Secretion*.

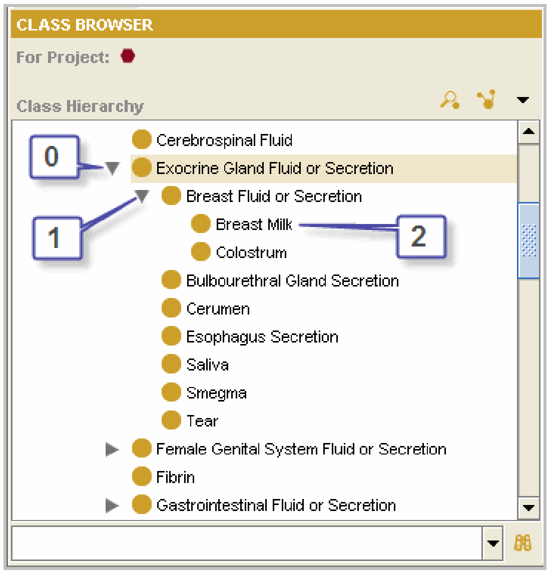
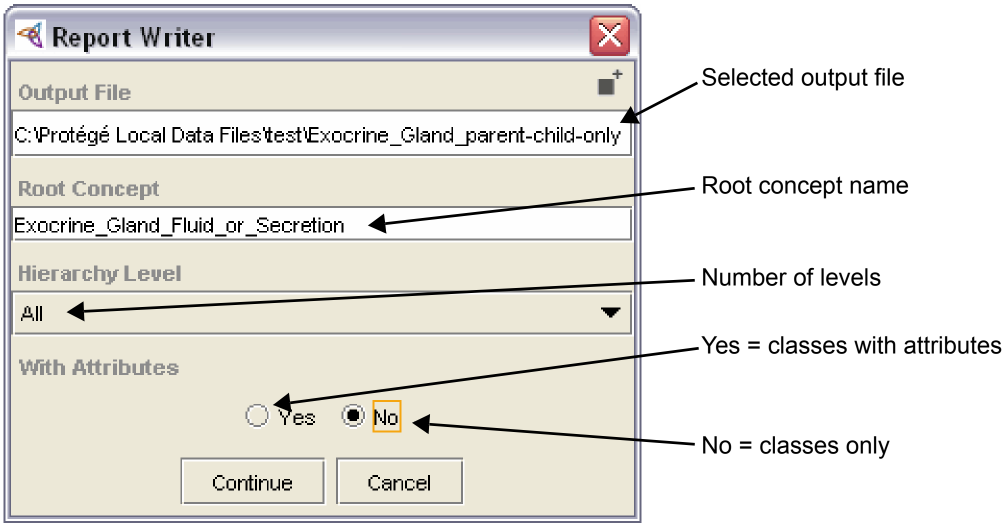


Figure 10.5 Level numbers for Exocrine Gland Fluid or Secretion

6. Select one of the options under the **With Attributes** section:

   - Select **Yes** to generate a report that includes the parent, all subclasses, and all properties and restrictions; or
   - Select **No** to include only the parent and all subclasses.

The example in *Figure 10.6* uses the **No** option.



Output File: C:\Protégé Local Data Files\test\Exocrine_Gland_parent-child-only

Root Concept: Exocrine_Gland_Fluid_or_Secretion

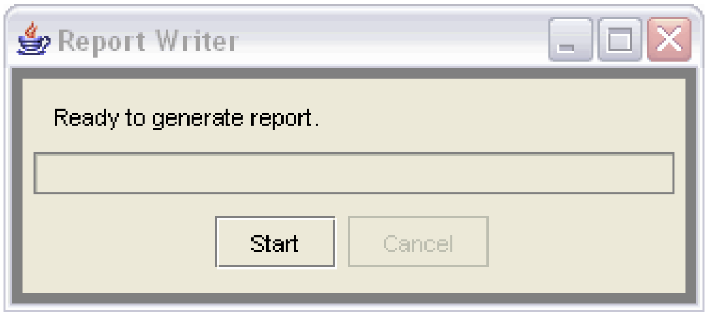
Hierarchy Level: All

With Attributes: ○ Yes ● No

Figure 10.6 Report Writer window with completed values

7. Click the **Continue** button.

   The Report Writer status window opens in the upper left corner of your computer screen (*Figure 10.7*).



Ready to generate report.

Figure 10.7 Report Writer status window as it initially appears

8. Click the **Start** button.

   The window displays a progress indicator bar.

9. When you see a *Report generation completed* message, click the standard Windows **Close** button in the upper right corner to close the window.

   Caution: Protégé opens a status window for each report that you generate, so if you generate ten reports, you will open ten windows. Remember to close the status window after each report is generated.

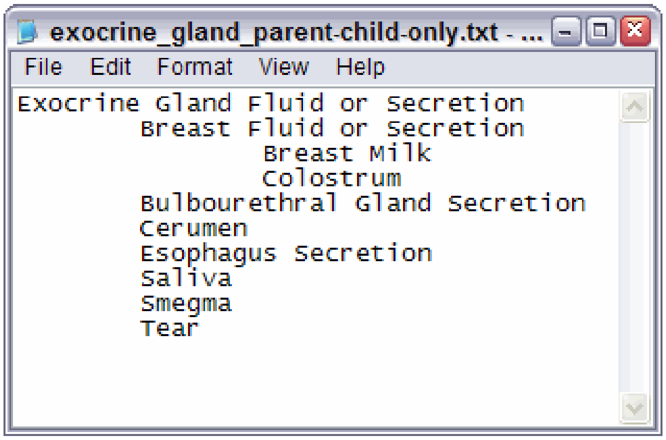
10. Using Windows Explorer, browse to the output file that you used for the report.

11. Open the file in a text editor such as Notepad.

12. Try generating several reports with and without attributes.

Sample Reports

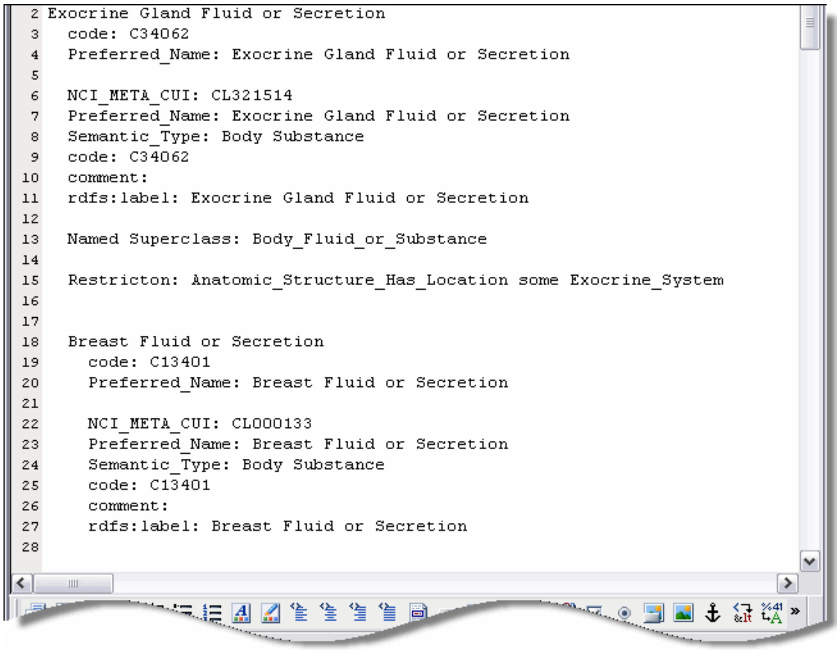
The report shown in *Figure 10.8* shows only classes and subclasses and excludes attributes. To create a similar report, select the **No** option in the With Attributes section of the Report Writer window.



```
Exocrine Gland Fluid or Secretion
        Breast Fluid or Secretion
                Breast Milk
                Colostrum
        Bulbourethral Gland Secretion
        Cerumen
        Esophagus Secretion
        Saliva
        Smegma
        Tear
```

Figure 10.8 Report with classes only - no attributes

This second sample report shows classes, subclasses, properties, and restrictions. To create a similar report, select the **Yes** option in the With Attributes section of the Report Writer window.



```
Exocrine Gland Fluid or Secretion
  code: C34062
  Preferred_Name: Exocrine Gland Fluid or Secretion

  NCI_META_CUI: CL321514
  Preferred_Name: Exocrine Gland Fluid or Secretion
  Semantic_Type: Body Substance
  code: C34062
  comment:
  rdfs:label: Exocrine Gland Fluid or Secretion

  Named Superclass: Body_Fluid_or_Substance

  Restricton: Anatomic_Structure_Has_Location some Exocrine_System


  Breast Fluid or Secretion
    code: C13401
    Preferred_Name: Breast Fluid or Secretion

    NCI_META_CUI: CL000133
    Preferred_Name: Breast Fluid or Secretion
    Semantic_Type: Body Substance
    code: C13401
    comment:
    rdfs:label: Breast Fluid or Secretion

```

Figure 10.9 Report with classes, subclasses, properties, and restrictions

Loading a Batch of Classes for Editing

Protégé enables you to load a batch of classes for editing. This is useful when you want to import a large number of classes and then edit them in the Protégé environment.

About the Input File Format for a Batch Load

You first create the batch file by specifying the following basic data for each class. Each of the three fields is case-sensitive:

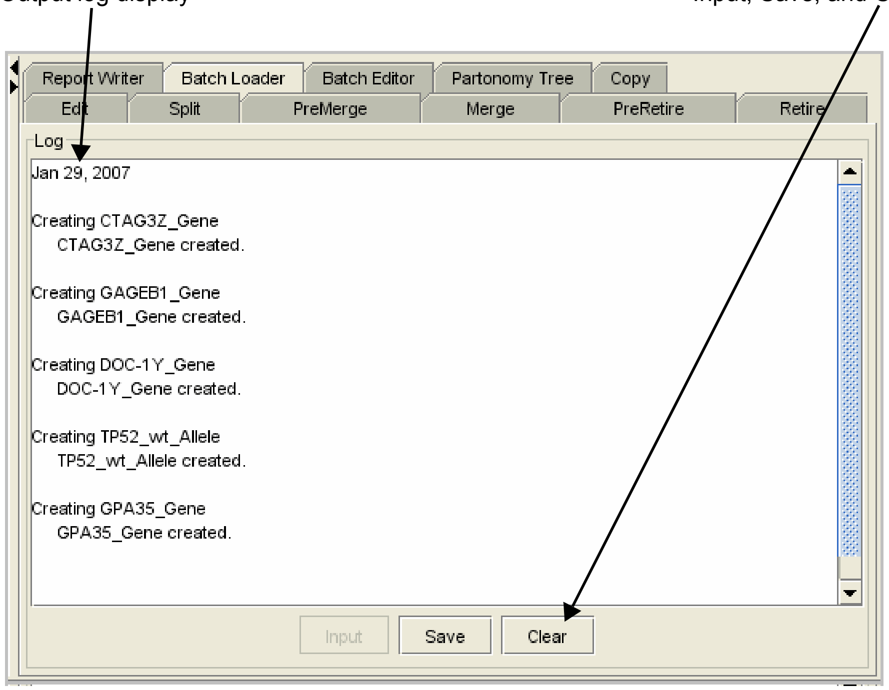
- **Field 1:** The name of the class (with underscores)
- **Field 2:** The preferred name of the class
- **Field 3:** The name of the parent class.

The batch-loading process accepts a tab-delimited (text) input file using the specified format. You can set up input file in a text editor such as Notepad or in a spreadsheet program such as Excel. In either case, save the file as a tab-delimited text file (`.txt`).

Note: If you need to load more than 200 classes, you must send your input file to a workflow manager. The manager will load the classes on Friday during the database comparison (Prompt). Managers do not load batches of classes on the last Friday of the month because that day is reserved for updates to the database. Batches of more than 200 classes that are submitted for the last Friday of the month will be considered on a case-by-case basis.

About the Batch Loader Subtab

The Batch Loader subtab (*Figure 10.10*) manages the process of importing a batch of classes. The Log area is empty until you specify and load an input file.


Output log display

Input, Save, and Clear buttons

```
Jan 29, 2007

Creating CTAG3Z_Gene
    CTAG3Z_Gene created.

Creating GAGEB1_Gene
    GAGEB1_Gene created.

Creating DOC-1Y_Gene
    DOC-1Y_Gene created.

Creating TP52_wt_Allele
    TP52_wt_Allele created.

Creating GPA35_Gene
    GPA35_Gene created.
```

Figure 10.10 Batch Loader subtab

Table 10.1 lists the features of the Batch Loader subtab.

Feature	***Description***
Output log display	Shows the log status (the status of the batch load). If the load is successful, the contents of the output log show here. If the input file contains errors, such as entries that are already in the knowledge base, an error message shows here. If you need to start over, click the **Clear** button.
Input button	Opens a window in which you can specify an input file and an output log file.
Save button	Saves the log to an ASCII file and confirms the file creation.
Clear button	Clears the display area.

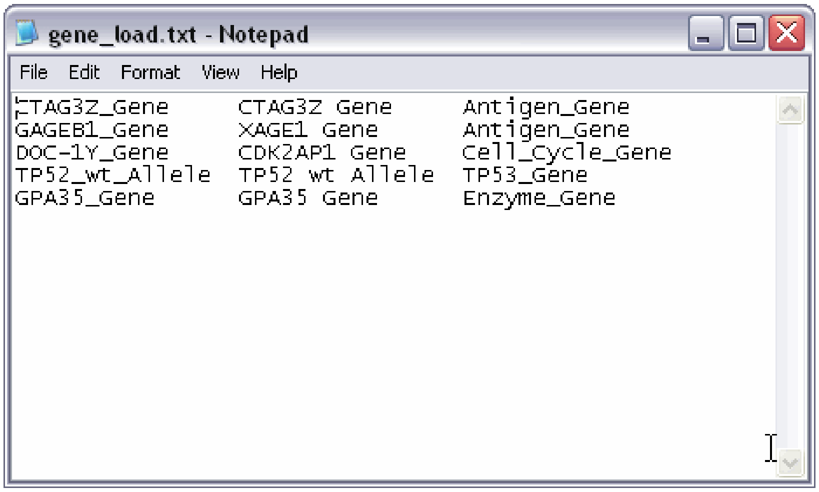
Table 10.1 Batch Loader subtab features

Using the Batch Loader Subtab to Load an Input File

To load a batch of classes, follow these steps:

1. Using a text editor such as Notepad, prepare and save a tab-delimited input file such as the example shown in *Figure 10.11*.

   Make sure that the file contains the class name, preferred name, and name of the parent class, as specified on page 150.



```
CTAG3Z_Gene	CTAG3Z Gene	Antigen_Gene
GAGEB1_Gene	XAGE1 Gene	Antigen_Gene
DOC-1Y_Gene	CDK2AP1 Gene	Cell_Cycle_Gene
TP52_wt_Allele	TP52 wt Allele	TP53_Gene
GPA35_Gene	GPA35 Gene	Enzyme_Gene
```

Figure 10.11 Tab-delimited input file for batch load

2. Select any class in the Class Browser on the left except *owl:Thing*.

   Note: In this step, the class you select has no bearing on the batch load procedure. You are performing this step because you cannot use the Batch Loader subtab unless something other than *owl:Thing* is selected in the Class Browser.

3. Click the **Batch Loader** subtab.

The subtab is initially blank, as shown in *Figure 10.12*. Once you set up and run the batch load process, the display area of the subtab shows the output of your batch load file.

4. Click the **Input** button at the bottom of the subtab.

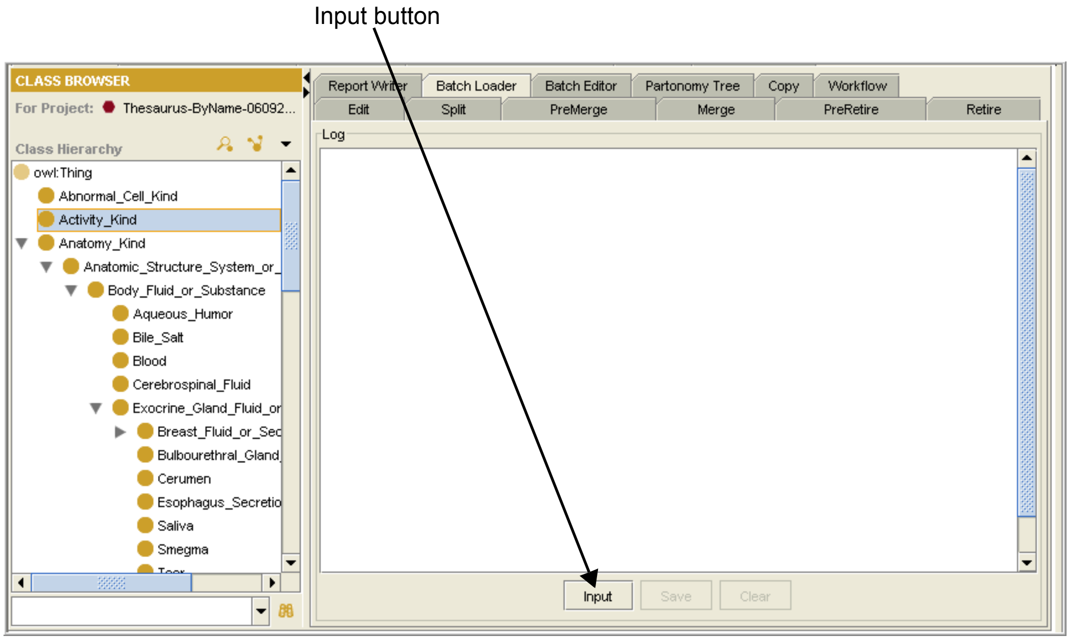


Figure 10.12 Batch Loader subtab

The Batch Loader window opens (*Figure 10.13*).

5. Browse for input and output log files by following these steps:

   a. Click the **Browse for File** button to the right of the **Input File** field .

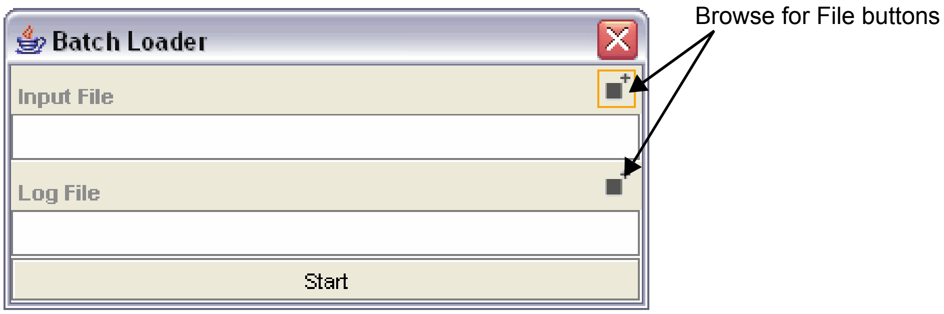
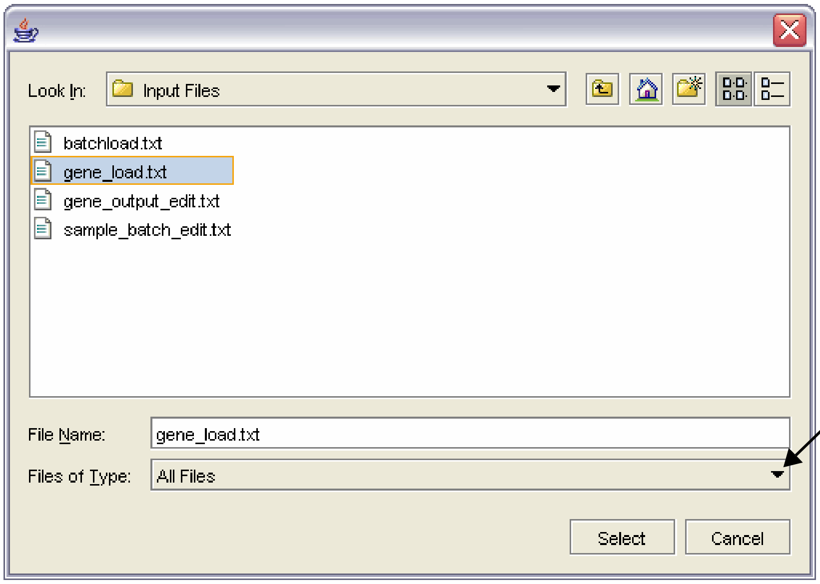


Figure 10.13 Batch Loader window

The browse window opens (*Figure 10.14* on page 153).

b. Click the arrow to the right of the **Files of Type** field, then select **All Files** from the drop-down list.



File Name: gene_load.txt

Files of Type: All Files

Set Files of Type to All Files.

Figure 10.14 Browse window for batch load files

c. Browse for an input file.

d. Click **Select** to close the browse window.

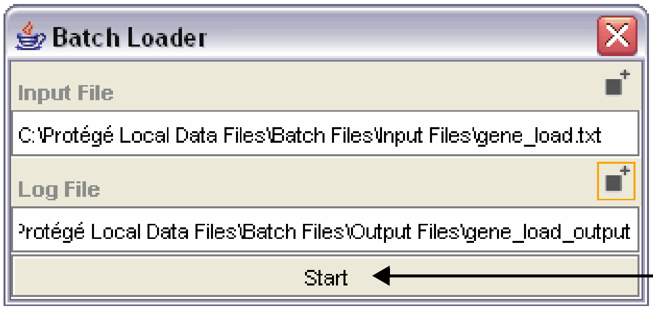
The input file path and name appear in the Input File field.

e. Repeat steps 5a through 5d to browse for an output log file.

Note: If you have not already created an output file, you can browse to a specific directory and then type a new name in the **File Name** field. Be sure to append the .txt extension to the name.

6. In the Batch Loader window, ensure that both the **Input File** and **Log File** fields show a file path and name (*Figure 10.15*).

7. Click the **Start** button in the bottom of the window.



Input File: C:\Protégé Local Data Files\Batch Files\Input Files\gene_load.txt

Log File: ›rotégé Local Data Files\Batch Files\Output Files\gene_load_output

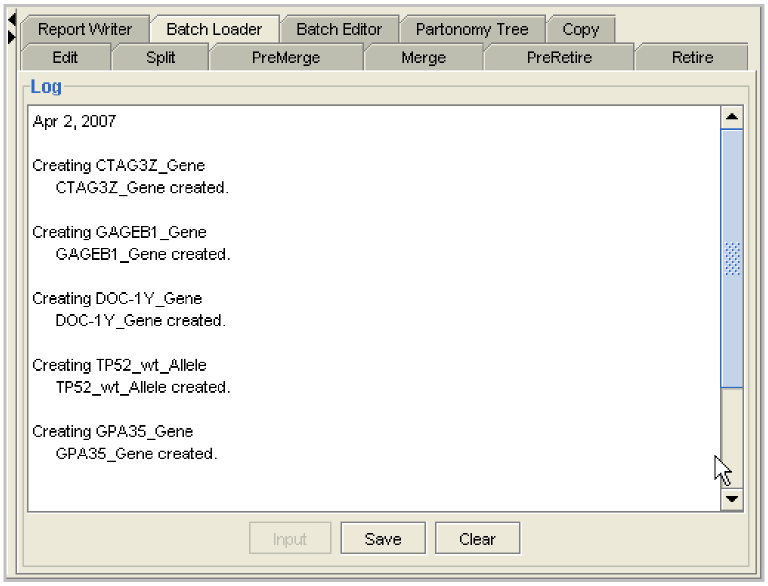
Start button

Figure 10.15 Batch Loader window with completed file paths and names

A progress bar confirms that the batch load has begun. When the process is finished, a message window confirms the number of completed actions.

8. Close the message window.

   The names of all newly loaded classes appear in the Batch Loader display area (*Figure 10.16*).



```
Apr 2, 2007

Creating CTAG3Z_Gene
    CTAG3Z_Gene created.

Creating GAGEB1_Gene
    GAGEB1_Gene created.

Creating DOC-1Y_Gene
    DOC-1Y_Gene created.

Creating TP52_wt_Allele
    TP52_wt_Allele created.

Creating GPA35_Gene
    GPA35_Gene created.
```

Figure 10.16 Newly imported batch load

9. To find the new classes in the Class Hierarchy, expand the branch for the parent class used in the batch load file.

Note: Although the **Save** button at the bottom of the Batch Loader subtab is still active, you do not need to click the button to save the new classes. They have already been saved to the database. You can, however, save the batch output to an external file as a record of the classes that you loaded.

Loading a Batch of Edited Classes

Batch editing is useful when you have a large number of classes and prefer to edit them outside of Protégé. After editing the classes in an external application such as Notepad or Excel, you can import the edited classes into Protégé.

Note: If you need to load more than 200 classes, you must send your input file to a workflow manager. The manager will load the classes on Friday during the database comparison (Prompt). Managers do not load batches of classes on the last Friday of the month because that day is reserved for updates to the database. Batches of more than 200 classes that are submitted for the last Friday of the month will be considered on a case-by-case basis.

About the Input File Format for a Batch Edit

The application that you use to edit a batch of classes should be capable of saving the edited file in tab-delimited text format (*.txt*). Once you have a text version, you are ready to import the file into Protégé.

The batch load file requires four to six fields, depending on what you are editing. *Table 10.2* describes the fields.

Field No.	***Accepted Values***	***Description***
1	Concept identifier (case sensitive)	For the current configuration, use a class name. In future configurations, this value may be a code.
2	new | edit | delete (case-sensitive)	Use one of these three accepted values to describe the action to be performed on the data. **Note:** If the value in Field 3 is *parent*, the value in this field must be **new** or **delete**.
3	property | role | parent | association	Use one of these four accepted values to describe the property or parent class being modified. **Note:** If the value in this field is *parent*, the value in Field 2 must be **new** or **delete**.
4	Property name or parent name	The value used here is determined by the value in Field 3.
5	See description.	This field is required for a new, edit, or delete action, except when the value in Field 3 is *parent*. For an edit action, this field stores the existing (unedited) value.
6	See description.	For an edit action, this field stores the new (edited) value. Otherwise, it is not required.

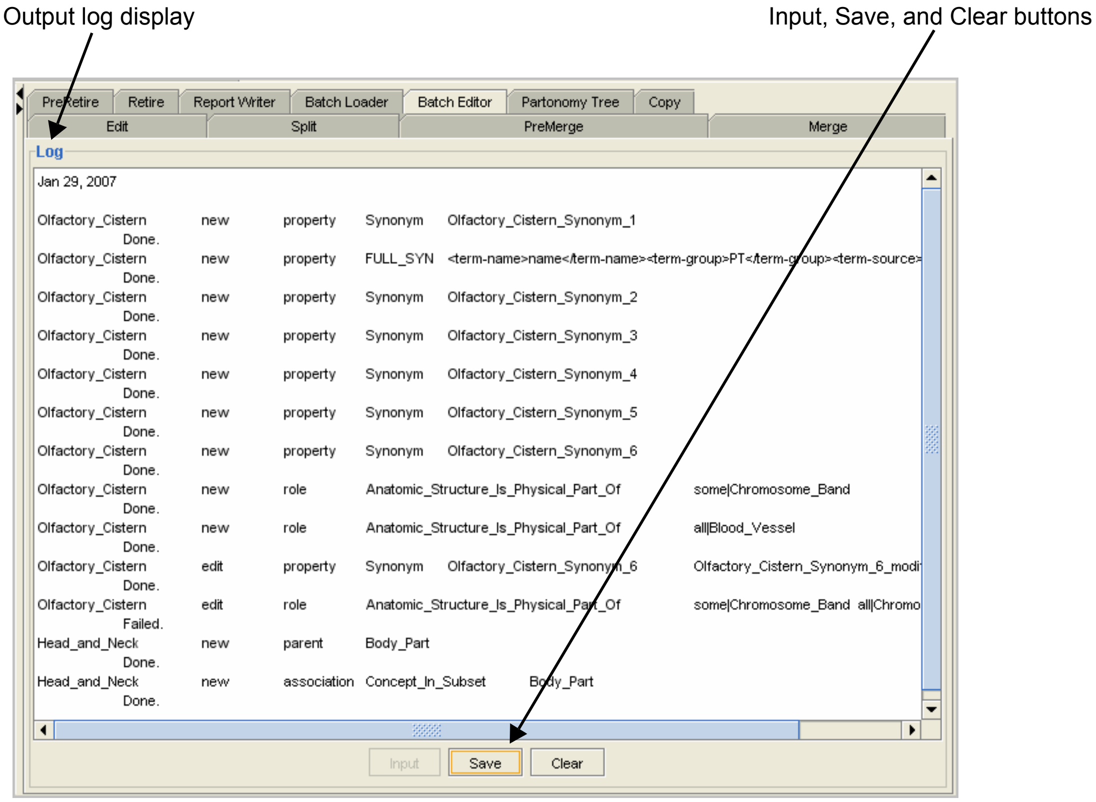
Table 10.2 Fields and values for a batch edit file

Note: Qualified properties such as FULL_SYN require that the XML tags of the sub-elements be included as a value. For example, a new FULL_SYN property for Olfactory_Cistern might show the following value in Field 5:

<term-name>name</term-name><term-group>PT</term-group>
<term-source>NCI</term-source>

About the Batch Editor Subtab

The Batch Editor subtab manages the process of importing batch-edit files. This tab is used to import a batch of edited classes and display the output. *Figure 10.17* shows the layout of this subtab.



```
Jan 29, 2007

Olfactory_Cistern        new      property    Synonym     Olfactory_Cistern_Synonym_1
                Done.
Olfactory_Cistern        new      property    FULL_SYN    <term-name>name</term-name><term-group>PT</term-group><term-source>
                Done.
Olfactory_Cistern        new      property    Synonym     Olfactory_Cistern_Synonym_2
                Done.
Olfactory_Cistern        new      property    Synonym     Olfactory_Cistern_Synonym_3
                Done.
Olfactory_Cistern        new      property    Synonym     Olfactory_Cistern_Synonym_4
                Done.
Olfactory_Cistern        new      property    Synonym     Olfactory_Cistern_Synonym_5
                Done.
Olfactory_Cistern        new      property    Synonym     Olfactory_Cistern_Synonym_6
                Done.
Olfactory_Cistern        new      role        Anatomic_Structure_Is_Physical_Part_Of        some|Chromosome_Band
                Done.
Olfactory_Cistern        new      role        Anatomic_Structure_Is_Physical_Part_Of        all|Blood_Vessel
                Done.
Olfactory_Cistern        edit     property    Synonym     Olfactory_Cistern_Synonym_6   Olfactory_Cistern_Synonym_6_modi
                Done.
Olfactory_Cistern        edit     role        Anatomic_Structure_Is_Physical_Part_Of        some|Chromosome_Band  all|Chromo
                Failed.
Head_and_Neck            new      parent      Body_Part
                Done.
Head_and_Neck            new      association Concept_In_Subset       Body_Part
                Done.
```

Figure 10.17 Batch Editor subtab

Table 10.3 describes the features of the Batch Editor subtab.

Feature	***Description***
Output log display	Shows the log status (the status of the batch edit load process).
Input button	Opens a window in which you can specify an input file and an output log file.
Save button	Saves the log to an ASCII file and confirms the file creation.
Clear button	Clears the display area.

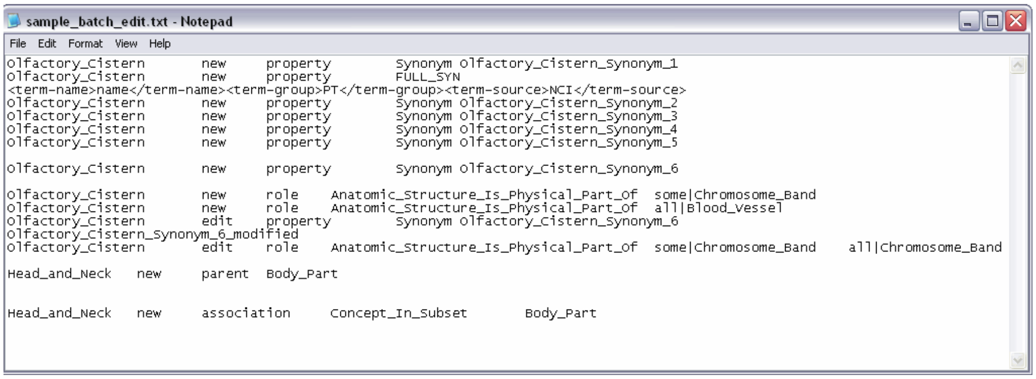
Table 10.3 Batch Editor subtab features

Using the Batch Editor Subtab to Import a Batch of Edited Classes

To import a batch of edited classes, follow these steps:

1. Using a text editor such as Notepad, prepare and save a tab-delimited input file such as the example shown in *Figure 10.18*.

   The file should be structured as described in *Table 10.2* on page 155.



```
olfactory_Cistern       new     property        Synonym olfactory_Cistern_Synonym_1
olfactory_Cistern       new     property        FULL_SYN
<term-name>name</term-name><term-group>PT</term-group><term-source>NCI</term-source>
olfactory_Cistern       new     property        Synonym olfactory_Cistern_Synonym_2
olfactory_Cistern       new     property        Synonym olfactory_Cistern_Synonym_3
olfactory_Cistern       new     property        Synonym olfactory_Cistern_Synonym_4
olfactory_Cistern       new     property        Synonym olfactory_Cistern_Synonym_5

olfactory_Cistern       new     property        Synonym olfactory_Cistern_Synonym_6

olfactory_Cistern       new     role    Anatomic_Structure_Is_Physical_Part_Of  some|Chromosome_Band
olfactory_Cistern       new     role    Anatomic_Structure_Is_Physical_Part_Of  all|Blood_Vessel
olfactory_Cistern       edit    property        Synonym olfactory_Cistern_Synonym_6
olfactory_Cistern_Synonym_6_modified
olfactory_Cistern       edit    role    Anatomic_Structure_Is_Physical_Part_Of  some|Chromosome_Band    all|Chromosome_Band

Head_and_Neck   new     parent  Body_Part


Head_and_Neck   new     association     Concept_In_Subset       Body_Part
```

Figure 10.18 Tab-delimited input file for batch load

2. With the NCI Edit subtab displayed, select any class in the Class Browser on the left except *owl:Thing*.

   Note: You are performing this step because you cannot use the Batch Editor subtab unless something other than *owl:Thing* is selected in the Class Browser.

3. Click the **Batch Editor** subtab.

   As shown in *Figure 10.19*, the subtab is initially blank until you set up and run the batch edit load process.

4. Click the **Input** button at the bottom of the subtab.

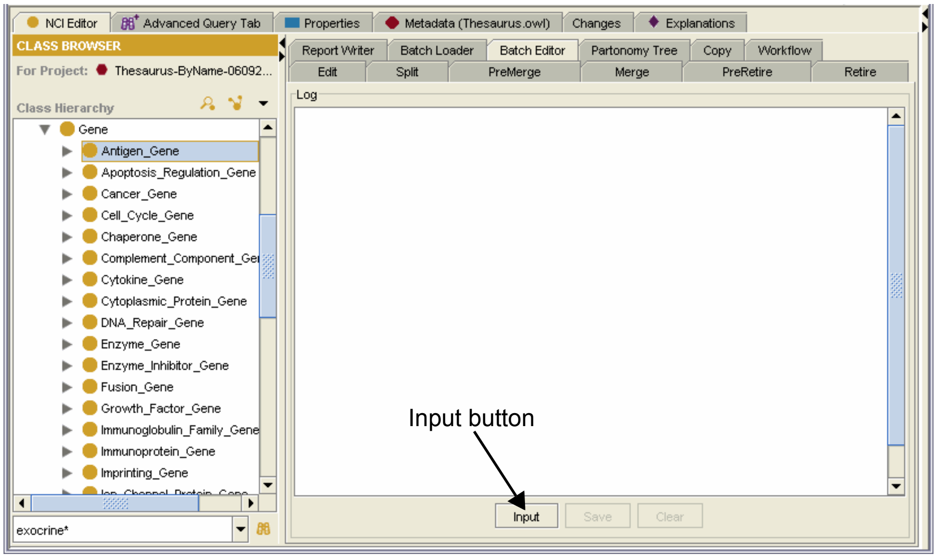


Figure 10.19 Batch Editor subtab

The Batch Editor window opens (*Figure 10.20*).

5. Browse for input and output log files by following these steps:

    a. Click the **Browse for File** button to the right of the Input File field .

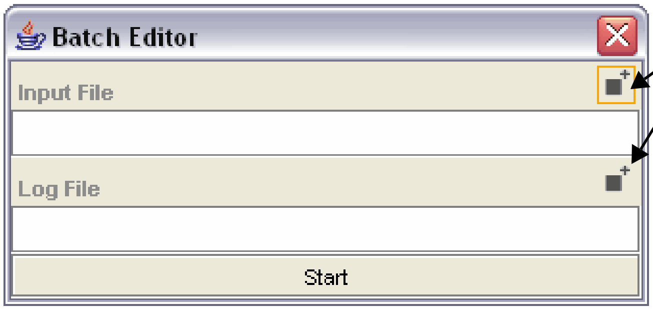
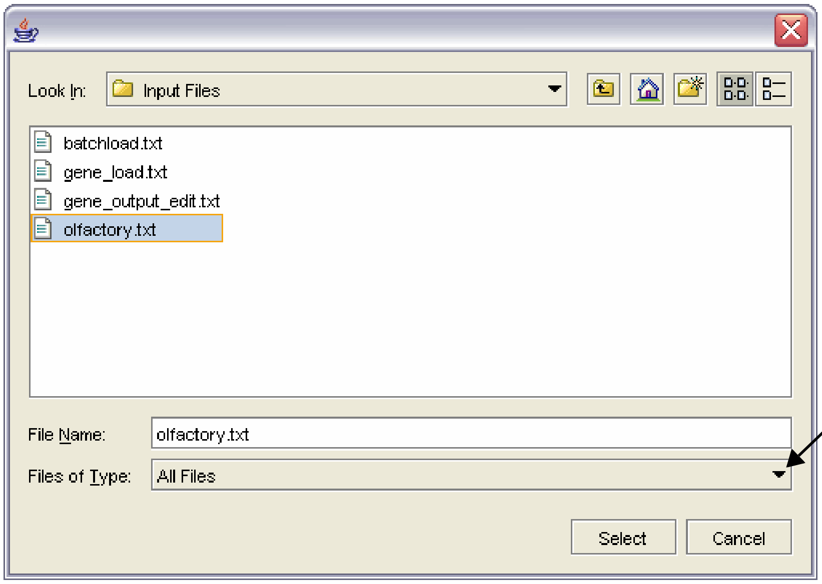


Figure 10.20 Batch Editor window

The browse window opens (*Figure 10.21*).

    b. Click the arrow to the right of the **Files of Type** field, then select **All Files** from the drop-down list.



File Name: olfactory.txt

Files of Type: All Files

Set Files of Type to All Files.

Figure 10.21 Browse window for batch edit files

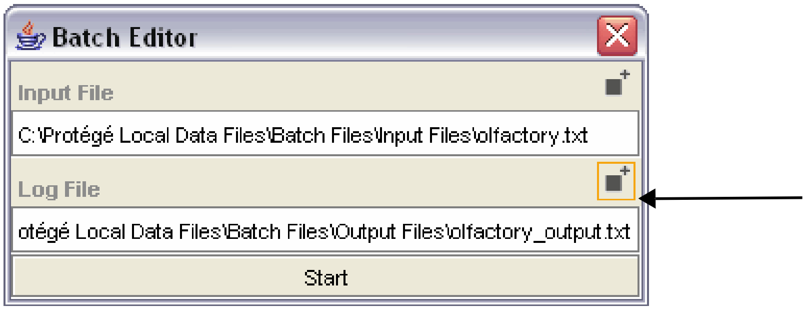
    c. Browse for an input file, then click **Select** to close the browse window.

    The input file path and name appear in the Input File field.

    d. Repeat steps 5a through 5d to browse for a log file.

Note: If you haven't already created an output file, you can browse to a specific directory and then type a new name in the **File Name** field. Be sure to append the .txt extension to the name.

6. In the Batch Editor window, ensure that both the **Input File** and **Log File** fields show a file path and name (*Figure 10.22*).



Input File: C:\Protégé Local Data Files\Batch Files\Input Files\olfactory.txt

Log File: otégé Local Data Files\Batch Files\Output Files\olfactory_output.txt

Start button

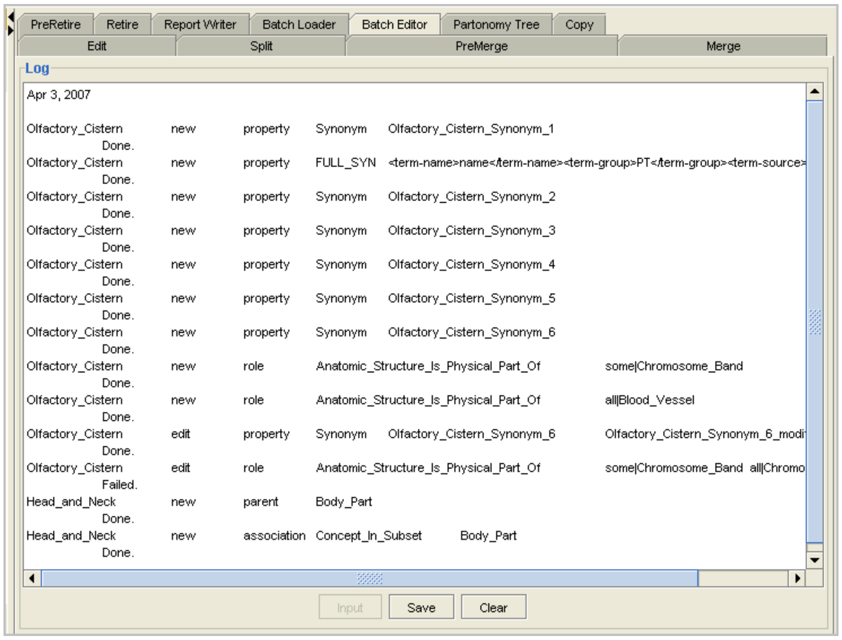
Figure 10.22 Batch Editor window with completed file paths and names

7. Click the **Start** button in the bottom of the Batch Editor window.

   When the process is finished, a message window confirms the number of completed actions.

8. Close the message window.

   The names of all newly loaded classes appear in the **Batch Loader** display area (*Figure 10.23*).



```
Apr 3, 2007

Olfactory_Cistern    new    property    Synonym    Olfactory_Cistern_Synonym_1
            Done.
Olfactory_Cistern    new    property    FULL_SYN   <term-name>name</term-name><term-group>PT</term-group><term-source>
            Done.
Olfactory_Cistern    new    property    Synonym    Olfactory_Cistern_Synonym_2
            Done.
Olfactory_Cistern    new    property    Synonym    Olfactory_Cistern_Synonym_3
            Done.
Olfactory_Cistern    new    property    Synonym    Olfactory_Cistern_Synonym_4
            Done.
Olfactory_Cistern    new    property    Synonym    Olfactory_Cistern_Synonym_5
            Done.
Olfactory_Cistern    new    property    Synonym    Olfactory_Cistern_Synonym_6
            Done.
Olfactory_Cistern    new    role        Anatomic_Structure_Is_Physical_Part_Of        some|Chromosome_Band
            Done.
Olfactory_Cistern    new    role        Anatomic_Structure_Is_Physical_Part_Of        all|Blood_Vessel
            Done.
Olfactory_Cistern    edit   property    Synonym    Olfactory_Cistern_Synonym_6        Olfactory_Cistern_Synonym_6_modi
            Done.
Olfactory_Cistern    edit   role        Anatomic_Structure_Is_Physical_Part_Of        some|Chromosome_Band  all|Chromo
            Failed.
Head_and_Neck        new    parent      Body_Part
            Done.
Head_and_Neck        new    association Concept_In_Subset      Body_Part
            Done.
```

Figure 10.23 Newly imported batch edit

9. To find the new classes in the Class Hierarchy, expand the branch for the parent class used in the batch load file.

Note: Although the **Save** button at the bottom of the Batch Loader subtab is still active, you do not need to click the button to save the newly edited classes. The changes have already been saved to the knowledge base. You can, however, save the batch output to an external file as a record of the batch edits that you loaded.

CHAPTER 11

WORKFLOW FOR NCI EDITORS

This chapter provides an overview of BiomedGT and the workflow processes and procedures used by external collaborators and NCI terminology curators. Although the procedures in this chapter are for NCI editors only, the overview of the workflow process includes stages for which a workflow manager has primary responsibility.

Topics in this Chapter

Overview of BiomedGT

The following topics introduce you to the BiomedGT terminology and the BiomedGT wiki.

About BiomedGT

This topic provides background information about the BiomedGT terminology and its intended audience.

Purpose

Biomedical terminologies are typically large, standalone terminologies that cover many domains in one structure and use a centralized editing model. Recognizing that this type of terminology no longer adequately supports the needs of the biomedical research community, the NCI Center for Bioinformatics (NCICB) and the cancer Biomedical Informatics Grid (caBIG™) have launched the Biomedical Grid Terminology (BiomedGT).

BiomedGT is an open, collaboratively developed terminology for translational research. This new terminology builds on the strengths of the NCI Thesaurus, including concept orientation, description logic, and public accessibility. While the initial content of

BiomedGT is based on the NCI Thesaurus, the terminology has been restructured to facilitate open content development. The goal is to evolve BiomedGT into a set of smaller federated sub-terminologies called *domain namespaces*, which cover specific areas of biomedical knowledge. As experts in the relevant research communities stabilize the content of the various domain namespaces, new content will be incorporated into the description logic-based, federated BiomedGT by the NCICB EVS and volunteer editors.

BiomedGT is intended to be the first of a new generation of biomedical terminologies that are created using an open content development approach. Such an approach empowers the wider biomedical research community to directly participate in extending and refining the terminology upon which they depend. The approach also enables maximum community participation without requiring that contributors learn complex terminology curation software and without sacrificing the quality assurance standards needed to manage biomedical terminologies.

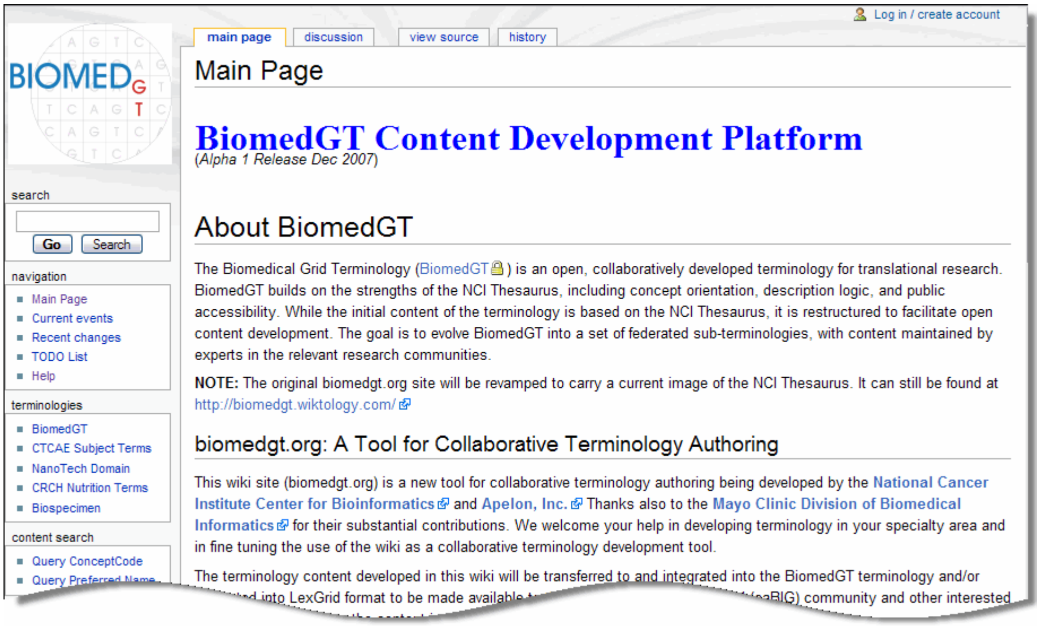
Audience

BiomedGT is intended to address the needs of biomedical researchers who work in both basic and clinical research. Researchers can acquire content by

- downloading it in multiple formats such as OWL and RRF;
- using Web-enabled applications; or
- using Application Programming Interfaces (APIs).

About the BiomedGT Wiki

The BiomedGT wiki (http://biomedgt.org) is a Web-based, collaborative terminology authoring tool being developed by the NCICB and Apelon, Inc.



Main Page

BiomedGT Content Development Platform
(Alpha 1 Release Dec 2007)

About BiomedGT

The Biomedical Grid Terminology (BiomedGT) is an open, collaboratively developed terminology for translational research. BiomedGT builds on the strengths of the NCI Thesaurus, including concept orientation, description logic, and public accessibility. While the initial content of the terminology is based on the NCI Thesaurus, it is restructured to facilitate open content development. The goal is to evolve BiomedGT into a set of federated sub-terminologies, with content maintained by experts in the relevant research communities.

NOTE: The original biomedgt.org site will be revamped to carry a current image of the NCI Thesaurus. It can still be found at http://biomedgt.wiktology.com/

biomedgt.org: A Tool for Collaborative Terminology Authoring

This wiki site (biomedgt.org) is a new tool for collaborative terminology authoring being developed by the **National Cancer Institute Center for Bioinformatics** and **Apelon, Inc.** Thanks also to the **Mayo Clinic Division of Biomedical Informatics** for their substantial contributions. We welcome your help in developing terminology in your specialty area and in fine tuning the use of the wiki as a collaborative terminology development tool.

The terminology content developed in this wiki will be transferred to and integrated into the BiomedGT terminology and/or

Figure 11.1 BiomedGT Home page

The BiomedGT wiki pulls terminology data from a read-only database and displays it on a Web page. This facilitates the process described in *Table 11.1*.

Stage	**Action**	**Responsibility**
1.	Review the read-only, published BiomedGT terminology.	Outside collaborator
2.	Using the provided Web form, submit a *structured* or *unstructured* proposal for any of the following reasons: • To suggest changes to existing concepts (structured) • To suggest the addition of new concepts (structured) • To direct NCI curators to a specific wiki discussion page by providing the URL (unstructured). Structured proposals are also called *packages* because they often include batches of concepts. Unstructured proposals typically refer to a wiki discussion page that might discuss specifics about a concept and then make a recommendation.	Outside collaborator
3.	Import packages and incorporate them into the vocabulary as needed. This stage has its own internal workflow processes and procedures as described in *Overview of Internal NCI Workflow* on page 164.	NCI BiomedGT curators
4.	Communicate with submitting collaborators to clarify suggestions or explain the outcome of decisions regarding proposals.	NCI BiomedGT curators
5.	Publish updated read-only database and display it on the Web page.	BiomedGT administrator

Table 11.1 Workflow process: BiomedGT to Protégé

For more information about BiomedGT, see the following online resources:

- BiomedGT project on GForge:
  http://gforge.nci.nih.gov/projects/biomedgt/
- EVS section of the NCICB web site:
  http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary
- BiomedGT wiki site:
  http://biomedgt.org

Overview of Internal NCI Workflow

As biomedical research community experts increasingly use the BiomedGT wiki for collaborative terminology development, NCI curators need a managed, organized workflow process. This topic discusses the process established for the NCI.

Primary Goals

Following are the primary goals of the BiomedGT/Protégé workflow:

- To provide capability that supports workflow manager and modeler roles, including assignments, tracking, auditing, and reporting
- To provide a means for extracting new terms from the BiomedGT wiki and incorporating them into assignments for NCI Protégé modelers
- To provide reports that help management assess modeling progress.

Workflow Roles

The internal workflow for NCI Protégé consists of two primary roles:

- **Manager**

  A manager oversees the coordination and distribution of work assignments. The assignments may originate from the BiomedGT wiki or from other members of the editing team.

  The manager performs the following procedures:

  - Imports packages from the BiomedGT wiki
  - Creates assignments for editors
  - Tracks assignments
  - Incorporates completed assignments into the NCI Thesaurus.

  Note: The manager may also perform routine editing tasks.

- **Modeler (Editor)**

  A modeler is an NCI editor who performs routine editing tasks. Modelers cannot import packages, create assignments, or delete assignments. However, they can make *suggestions*, which are similar to assignments but which do not become assignments until a manager has reviewed them.

  The modeler performs the following procedures:

  - Receives assignments and accepts or rejects them
  - Incorporates new or changed terminology into the NCI Thesaurus using the NCI Protégé work environment
  - Maintains responsibility for the assigned concept, including any follow-up communication to the collaborator who submitted the concept for consideration
  - Marks assignments as complete and submits them to the manager.

Workflow Stages

Table 11.2 provides an overview of the internal workflow stages and the responsible role for each stage.

Note: Each submitted item is called a *workflow item* until it has been assigned to a modeler. It then becomes an *assignment*.

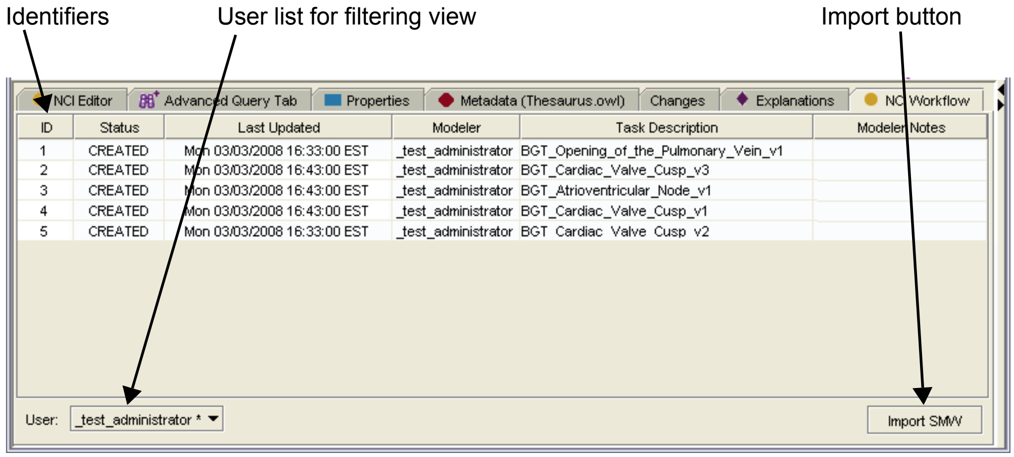
Stage	*Action*	*Responsibility*
1.	Log into Protégé.	Manager
2.	Using the NCI Workflow tab, import packages from the BiomedGT wiki into Protégé.	Manager
3.	Create workflow items with a status of CREATED.	System
4.	Review each new workflow item and assign it to a modeler.	Manager
5.	Change the status of the workflow item to ASSIGNED.	System
6.	Log into Protégé.	Modeler
7.	Check the NCI Workflow tab for assignments.	Modeler
8.	Accept an assignment.	Modeler
9.	Change the status of the assignment to ACCEPTED.	System
10.	Examine the proposed and existing (if any) concepts.	Modeler
11.	Perform one of the following procedures: ● Edit and save the existing concept (if any); if completed, mark the assignment as COMPLETED. ● Add notes to the assignment regarding the proposed concept and existing concept (if any); if completed, mark the assignment as COMPLETED. ● Reject the assignment.	Modeler
12.	Change the status of the assignment to UPDATED (if not marked COMPLETED), COMPLETED, or REJECTED, depending on the action in step 11.	System
13.	Review completed and rejected assignments; address rejected assignments according to the nature of the proposal.	Manager

Table 11.2 BiomedGT wiki process for curating terminology

In addition to work items generated by BiomedGT wiki imports, the manager also handles work items that have been submitted internally by modelers. The system assigns a status of SUGGESTED to internally submitted work items.

About the NCI Workflow Tab

The NCI Workflow tab is the major workflow component used by both managers and modelers. The tab layout is a table grid format, with each row representing a workflow item or assignment (*Figure 11.2*).



ID	Status	Last Updated	Modeler	Task Description	Modeler Notes
1	CREATED	Mon 03/03/2008 16:33:00 EST	_test_administrator	BGT_Opening_of_the_Pulmonary_Vein_v1	
2	CREATED	Mon 03/03/2008 16:43:00 EST	_test_administrator	BGT_Cardiac_Valve_Cusp_v3	
3	CREATED	Mon 03/03/2008 16:43:00 EST	_test_administrator	BGT_Atrioventricular_Node_v1	
4	CREATED	Mon 03/03/2008 16:43:00 EST	_test_administrator	BGT_Cardiac_Valve_Cusp_v1	
5	CREATED	Mon 03/03/2008 16:33:00 EST	_test_administrator	BGT Cardiac Valve Cusp v2	

Figure 11.2 NCI Workflow tab

Protégé determines what the NCI Workflow tab displays based on the privileges of the user who is currently logged in:

- Managers can see all assigned and unassigned workflow items and can identify the assignee and status of each assigned item. They can also filter the assignments by user.
- Modelers see only workflow items that have been assigned to them.

Table 11.3 describes the fields that identify each workflow item and assignment.

Field	***Description***
ID	Shows a sequential number that is assigned to each workflow item. Once a number is assigned to an item, it remains the same, even if the same item is re-imported.
Status	Shows the current status of a workflow item or assignment: • CREATED: The manager has imported a workflow item but has not yet assigned it. • SUGGESTED: A modeler has created a workflow item that is relevant to an existing concept. • ASSIGNED: The manager has assigned a workflow item to a modeler but the modeler has not yet accepted it. • ACCEPTED: The modeler has accepted the assignment. • UPDATED: The modeler saved a change to a class or added a note but did not mark the assignment as COMPLETED. • COMPLETED: The modeler has completed the assignment.

Table 11.3 NCI Workflow tab fields

Field	**Description**
Last Updated	Shows the weekday, date, and time in the following format: Fri 03/07/2008 16:34:00 EST
Modeler	Initially shows the name of the user (manager) who imported the workflow item. Once assigned, this field displays the name of the assignee.
Task Description	For structured proposals, shows the name of a concept or a URL pointing to a wiki discussion page. Concept names are preceded by the letters BGT to note that the concept has been imported from the BiomedGT wiki.
Modeler Notes	Shows the most recent note added to a workflow item or assignment, followed by an ellipsis (...). If you hover the mouse pointer over the date and time, pop-up text shows the note text starting with the most recent entry.

Table 11.3 NCI Workflow tab fields (Continued)

The grid area of the NCI Workflow tab supports right-click commands. *Table 11.4* describes the available commands. Some commands may appear as unavailable depending on your user privileges.

Command	**Description**
Edit Concept	Displays the proposed concept and existing concept (if any) on the Workflow subtab. You can then edit, add notes to, or reject the concept.
Accept Assignment	Enables you to accept an assignment.
Modify Assignment	Opens a window in which you can add notes and view previous notes.
Complete Assignment	Enables you to change the status of an assignment to COMPLETED.
Delete Assignment	Enables you to delete an assignment, depending on your user privileges and the status of the assignment: ● Managers can delete assignments with a status of CREATED, SUGGESTED, REJECTED, or COMPLETED. ● Modelers can only delete SUGGESTED assignments.
Cancel	Closes the shortcut menu.

Table 11.4 NCI Workflow tab - right-click commands

Workflow Procedures

The following topics detail the steps for the main procedures that NCI editors (modelers) typically perform. Most of the procedures originate from the NCI Workflow tab.

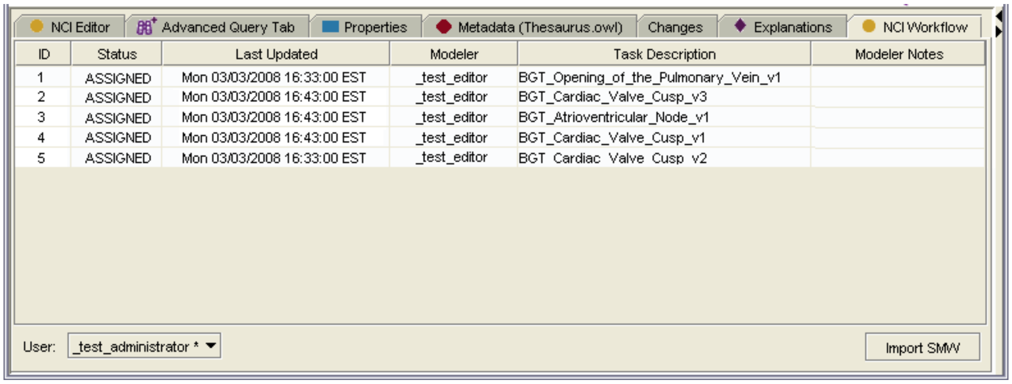
Note: For more information about modeler procedures in relation to the overall NCI internal workflow process, see *Overview of Internal NCI Workflow* on page 164.

- *Accepting a Workflow Assignment* on this page
- *Editing an Assigned Concept* on page 169
- *Adding Notes to an Assignment* on page 170
- *Rejecting an Assignment* on page 171
- *Completing an Assignment* on page 172
- *Suggesting an Assignment* on page 172

Accepting a Workflow Assignment

Before you can work on an assignment, you must first accept it. To accept an assignment, follow these steps:

1. Log in to Protégé.
2. Click the **NCI Workflow** tab.
3. In the list of workflow items, look for new items with a status of ASSIGNED (*Figure 11.3*).



ID	Status	Last Updated	Modeler	Task Description	Modeler Notes
1	ASSIGNED	Mon 03/03/2008 16:33:00 EST	_test_editor	BGT_Opening_of_the_Pulmonary_Vein_v1	
2	ASSIGNED	Mon 03/03/2008 16:43:00 EST	_test_editor	BGT_Cardiac_Valve_Cusp_v3	
3	ASSIGNED	Mon 03/03/2008 16:43:00 EST	_test_editor	BGT_Atrioventricular_Node_v1	
4	ASSIGNED	Mon 03/03/2008 16:43:00 EST	_test_editor	BGT_Cardiac_Valve_Cusp_v1	
5	ASSIGNED	Mon 03/03/2008 16:33:00 EST	_test_editor	BGT Cardiac Valve Cusp v2	

Figure 11.3 Workflow items with ASSIGNED status

4. Right-click on the row that displays the assignment you want to accept.
5. In the shortcut menu, select **Accept Assignment**.

   The Status column now shows a status of ACCEPTED.

Tip: You can always use the NCI Workflow tab to periodically check for new workflow assignments and verify the status of your existing assignments.

Editing an Assigned Concept

Prerequisite procedure: *Accepting a Workflow Assignment* on page 168

After accepting an assignment, you need easy access to the related concepts. Regardless of whether you are working on an assignment for the first time or returning to an assignment that you have not yet completed, you can access the related concepts from the NCI Workflow tab.

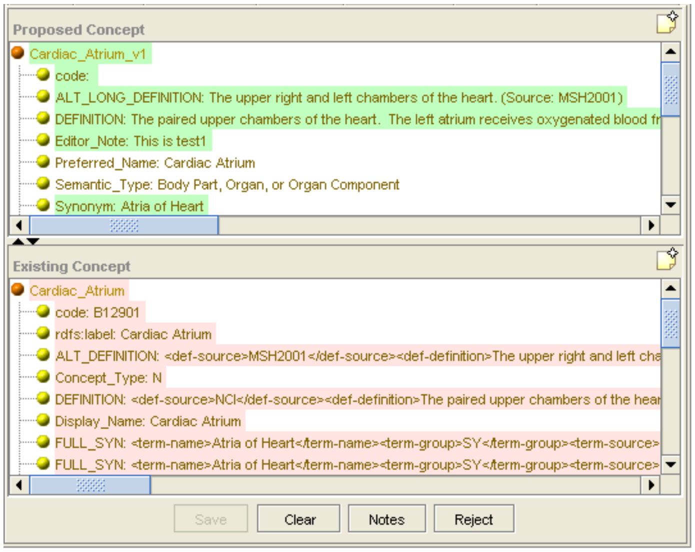
To begin work or resume work on an assignment, follow these steps:

1. Click the **NCI Workflow** tab, then locate the assignment requiring your attention.
2. Right-click anywhere on the row that displays the assignment.
3. In the shortcut menu, click **Edit Concept**.

   The result depends on the origin of the assignment, as described in *Table 11.5*.

Assignment Origin	***Result***
Imported wiki package with proposed new concept or proposed changes to an existing concept...	Protégé displays the Workflow subtab with the *proposed* concept in the upper pane and the *existing* concept (if any) in the lower pane (*Figure 11.4*).
Imported wiki package with reference to a wiki discussion page...	A pop-up message window displays the URL of the discussion page. Eventually, the page content will be displayed within the Protégé work environment.
Suggested by an NCI editor...	Protégé displays the Edit subtab > Basic Data subtab with details for the existing concept.

Table 11.5 Results of selecting different types of assignments



Cardiac_Atrium_v1
- code:
- ALT_LONG_DEFINITION: The upper right and left chambers of the heart. (Source: MSH2001)
- DEFINITION: The paired upper chambers of the heart. The left atrium receives oxygenated blood fr
- Editor_Note: This is test1
- Preferred_Name: Cardiac Atrium
- Semantic_Type: Body Part, Organ, or Organ Component
- Synonym: Atria of Heart

Cardiac_Atrium
- code: B12901
- rdfs:label: Cardiac Atrium
- ALT_DEFINITION: <def-source>MSH2001</def-source><def-definition>The upper right and left cha
- Concept_Type: N
- DEFINITION: <def-source>NCI</def-source><def-definition>The paired upper chambers of the hear
- Display_Name: Cardiac Atrium
- FULL_SYN: <term-name>Atria of Heart</term-name><term-group>SY</term-group><term-source>
- FULL_SYN: <term-name>Atria of Heart</term-name><term-group>SY</term-group><term-source>

Figure 11.4 Assignment from wiki import - proposed and existing concepts

Adding Notes to an Assignment

Prerequisite procedure: *Accepting a Workflow Assignment* on page 168

You can add notes to an assignment using both the NCI Workflow tab and the Workflow subtab. This topic explains both methods.

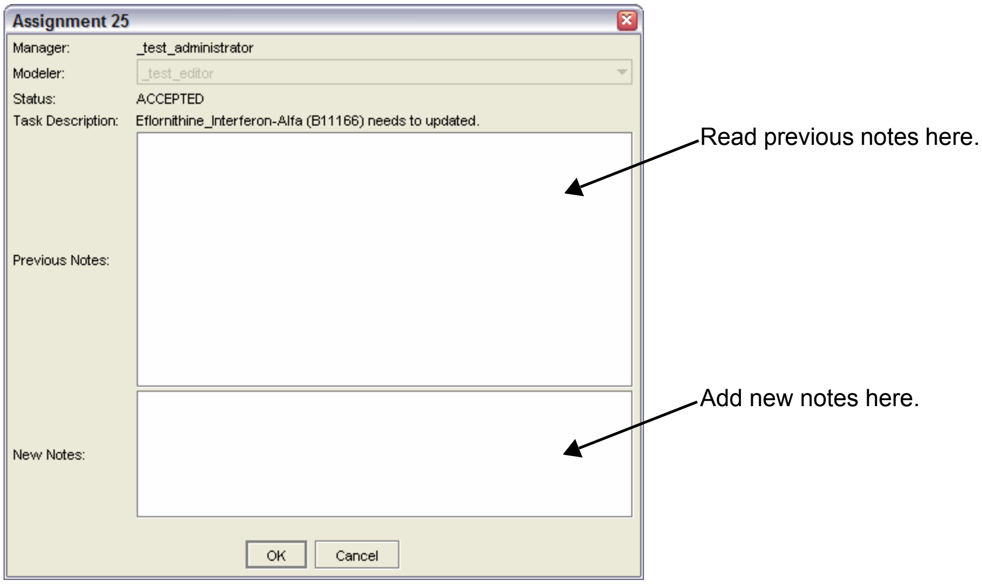
Using the NCI Workflow Tab

Protégé provides a window in which you can add notes to an assignment or read previous notes that you or others have already added.

To add notes to an assignment, follow these steps:

1. Click the **NCI Workflow** tab, then locate the assignment requiring notes.
2. Right-click anywhere on the row that displays the assignment.
3. In the shortcut menu, click **Modify Assignment**.

   A window opens with the assignment number in the title bar. The top pane shows any previously added notes, with the most recent note at the top. The bottom pane enables you to add a new note (*Figure 11.5*).



Assignment 25

Manager: _test_administrator

Modeler: _test_editor

Status: ACCEPTED

Task Description: Eflornithine_Interferon-Alfa (B11166) needs to updated.

Previous Notes:

New Notes:

Read previous notes here.

Add new notes here.

Figure 11.5 Window for adding notes to an assignment

4. Click **OK** to close the notes window.

Using the Workflow Subtab

Prerequisite procedure: *Editing an Assigned Concept* on page 169

To add notes while viewing proposed and existing concepts on the Workflow subtab, follow these steps:

1. Click the **Notes** button at the bottom of the subtab.

   The Modelers Notes window opens. The top pane shows any previously added notes, and the bottom pane enables you to add a new note.

2. Once you finish adding new notes, click **OK**.

Rejecting an Assignment

You may sometimes need to reject an assignment in cases such as the following:

- The proposed new concept already exists but was not found by the wiki user.
- The concept is deemed to be out of scope.

Note: Before rejecting an assignment, you or your workflow manager need to discuss the proposed new concept or concept changes with the outside collaborators who submitted the proposal. For example, if the requested modeling does not classify correctly, you may need to clarify the intended usage. You may find after asking for clarification that the concept can be modified to conform to NCI editing guidelines.

Prerequisite procedures:

- *Accepting a Workflow Assignment* on page 168
- *Editing an Assigned Concept* on page 169

If you cannot complete an assignment, or if the assignment should be assigned to someone else, you can reject that assignment.

To reject an assignment, follow these steps:

1. Ensure that the assignment is open for editing on the **Workflow** subtab.

   The subtab displays the proposed concept and the existing concept (if any).

2. Click the **Reject** button at the bottom of the subtab.

   The Rejection Notes window opens.

3. (Required) Add notes to the **New Notes** section (lower section) of the window.
4. To verify that the assignment status has been changed, click the **NCI Workflow** tab.

   The assignment now shows a REJECTED status.

Note: In the current release, you can reject only assignments that originate from a wiki package and include a proposed new concept or proposed changes to an existing concept. You cannot reject assignments that (1) originate from a wiki page and refer to a wiki discussion page or (2) originate as internal suggestions.

Completing an Assignment

Prerequisite procedures:

- *Accepting a Workflow Assignment* on page 168
- *Editing an Assigned Concept* on page 169

Once you have completed an assignment, you need to mark it as completed.

To mark an assignment as completed, follow these steps:

1. Click the **NCI Workflow** tab, then locate the assignment that you have completed.
2. Right-click anywhere on the row that displays the assignment.
3. In the shortcut menu, click **Complete Assignment**.

   The Status column now shows a status of COMPLETED.

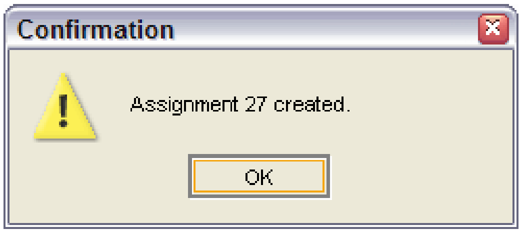
Suggesting an Assignment

You can suggest an assignment that is relevant to an existing concept. NCI Protégé creates a new workflow item with a status of SUGGESTED. The workflow manager can then assign the item to a modeler.

To suggest an assignment, follow these steps:

1. Select the **Copy** subtab.
2. Using the Class Browser, locate the class for which you want to suggest an assignment.
3. Drag the class into the upper pane of the **Copy** subtab.
4. Right-click any property in the upper pane, then select **Create Workflow Task** from the shortcut menu.

   A message window confirms that a new assignment has been created. The message includes the assignment number (*Figure 11.6*).



Assignment 27 created.

Figure 11.6 Message confirming creation of suggested assignment

5. To verify that the assignment has been created, click the **NCI Workflow** tab.
6. Locate the assignment by number.

   In the row for the new assignment, the Status column shows a status of SUGGESTED.

APPENDIX
A
TDE VS. PROTÉGÉ TERMINOLOGY

Table A.1 lists terms used in the Apelon Terminology Development Environment (TDE) and gives their equivalent terms in Protégé.

TDE Term	***Protégé Term or Description***
All	allValuesFrom (all, only, universal quantifier)
Association	AnnotationProperty of type ObjectProperty
Bottom	Nothing
Concept	• Class (set semantics) • Individuals
Defined	Necessary & Sufficient conditions asserted
Kind	Top-level disjoint class
Pick Lists	Enumerated Datatype AnnotationProperty
Primitive	Necessary conditions asserted
Property	AnnotationProperty (AnnotationProperty of type DatatypeProperty)
Qualifier	XML - as in TDE when we first began using it
Role	• ObjectProperty • DatatypeProperty
Role Expression	Restriction on property
Some	someValuesFrom (some, existential quantifier)
Top	Thing

Table A.1 TDE terms and their Protégé equivalents

GLOSSARY

This glossary defines acronyms, abbreviations, and terminology used in this guide .

Term	***Definition***
advanced query	A method of building a complex query using the Lucene Query tab; this type of search gives you more control over the search results.
anonymous class	A *class* used in assertions of *necessary* or *necessary and sufficient* conditions that describe or define the class and constrain the individuals that make up the class.
baseline comparison	A process whereby a workflow manager compares two versions of the Protégé database, examines the results, and resolves any conflicts.
batch editing	Editing a group of classes outside of Protégé and then importing them into the application.
batch loading	Importing a large number of classes and then editing them in the Protégé environment.
BiomedGT	An open, collaboratively developed terminology for translational research.
BiomedGT wiki	A Web-based, collaborative terminology authoring tool being developed by the NCICB and Apelon, Inc.
cardinality constraint	Specifies or limits the number of times that a property can be asserted on a class; cardinality is equal to a number, a minimum cardinality, or a maximum cardinality.
class	A *concept* in the Web Ontology Language (OWL); a class has a name, belongs to a namespace, and exists in relation to other classes.
classifier	See *reasoner*.
cloning	Process of creating a new class from an existing class, with the original class essentially serving as a template for the new class.
complex property	Property that provides both a value and additional information about a value, expressed using property qualifiers.
concept	A set of individuals in a given domain; in OWL, a *concept* is known as a class.
CTEP	Cancer Therapy Evaluation Program terminology; used in clinical trials administration.

Term	**Definition**
CTRM	Core anatomy, diagnosis, and agent terminology; used in Translational research by NCICB applications.
defined concept	Concepts with *necessary and sufficient* conditions; a defined concept is derived from a *primitive concept* using the constructors allowed by description logic, such as role assertions, intersections, and unions.
description logics (DL)	A family of languages that can be used to represent terminological systems; a DL is an extension of the *frame-based representation* formalism but with defined semantics based on set theory.
ELC2001	NCBI tissue taxonomy; used in tissue classification for genetic data such as cDNA libraries.
EVS	Enterprise Vocabulary Services; a partnership between the NCI Center for Bioinformatics and the NCI Office of Communications that works to harmonize and integrate the many diverse terminologies and coding frameworks used by the NCI and its partners.
first-order predicate logic	Complete first-order logic that enables us to make general statements about anonymous elements, with the introduction of variables as placeholders.
frame-based representation	The basic idea that important objects in our world fall into natural classes, and that all members of these classes share certain properties or attributes.
ICD03	Ontology classifications; used in cancer genome research and incidence reporting.
knowledge representation	Area of artificial intelligence (AI) research; this area asks how we can accurately encode the rich and highly detailed world of information that is required for the application area being modeled and yet, at the same time, capture the implicit common sense knowledge.
MDBCAC	Topology and morphology; used in cancer genome research.
medDRA	Regulatory reporting terminology; used in tissue classification for genetic data such as cDNA libraries.
merge	The process of combining synonymous classes, where the class meanings are not distinct from each other or are identical to an existing class; unless otherwise specified, the older class (the one with the lower code) survives and gains all of the roles, properties, parents, and children of the newer class. The newer class is retired.
MMHCC	Mouse Cancer Database terminology; used in Mouse Models of Human Cancer Consortium.
NCI Metathesaurus	A comprehensive biomedical terminology database that contains 1,100,000 concepts mapped to 2,500,000 terms with 5,000,000 relationships; based on the National Library of Medicine's Unified Medical Language System Metathesaurus (UMLS), the NCI Metathesaurus includes most UMLS terms and supplements them with additional cancer-centric vocabulary. The Metathesaurus excludes certain proprietary vocabularies and includes others with restricted use.
NCIPDQ	Expanded and re-organized PDQ; used for CancerLit indexing and clinical trials accrual.

Term	***Definition***
NCISEER	SEER terminology; used for incidence reporting.
NCI Source	Limited model of the NCI Thesaurus; used as a reference terminology for cancer research applications.
NCI Terminology Browser	An open-source NCI Terminology Web browser that provides direct access to a number of biomedical terminologies of special interest; in addition to the NCI Thesaurus, its terminologies include SNOMED CT, MedDRA, LOINC, VA NDF-RT, GO, and the MGED Ontology.
NCI Thesaurus	A core reference terminology and biomedical ontology. Implemented as a Description Logic vocabulary, the Thesaurus is a self-contained, logically consistent terminology providing rich textual and ontological descriptions of some 50,000 key biomedical concepts.
necessary and sufficient	Refers to a collection of asserted conditions that is necessary and sufficient to indicate whether an individual is a member of a class; this means that equivalent classes have the same exact set of individuals.
OWL	Web Ontology Language, designed to facilitate the semantic aspect of the Semantic Web; OWL is built on top of RDF/RDFS, an XML-based data representation scheme.
partonomy tree	Hierarchical representation showing classes that are connected by *part_of* relations.
pre-merge	Process whereby an editor flags classes for a merge; a workflow manager then approves and completes the merge.
pre-retire	Process whereby an editor flags a class for retirement; a workflow manager then approves the retirement and retires the class.
primitive concept	Concepts asserted with a *necessary* description.
Prompt plug-in	A tool used to run a *baseline comparison*.
property	A class attribute; see also *simple property* and *complex property*.
property restriction	See *role*.
reasoner	A tool that examines the relationships in the class hierarchy, computes the subclass relations between every named class, and determines whether possible modeling errors exist.
retire	Process of changing the status of a class that is no longer needed so that it appears as deprecated (preceded with a letter D) in the Class Hierarchy.
role	Binary relations between concepts; note that a qualified role assertion on a class defines an anonymous concept (the unnamed set of individuals making up the class—the DL view of concept). In OWL, a role is called a *property restriction*.
Semantic Web	An evolving extension of the World Wide Web in which the semantics of information and services on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content (from http://en.wikipedia.org).
simple property	A property that provides a text string value.

Term	***Definition***
simple search	Executing a search by typing a search string in a text field and then clicking a button to execute the search.
split	Process of creating a new class from another class and moving the appropriate atoms to the new class; the new class becomes the sibling of the original class and automatically inherits all of the subclasses of the original class.
URL	Uniform Resource Locator; the address of a resource on the Web.
value constraint	Places restrictions on the range of a property when applied to a class; examples include *some*, *all*, and *has value*.
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

INDEX