

# NCI EDITOR'S GUIDE TO USING PROTÉGÉ

*Version 1.1*



NATIONAL<sup>®</sup>  
CANCER  
INSTITUTE

Center for Bioinformatics

September 22, 2007



# CONTENTS

<b>List of Figures .....</b>	<b>vii</b>
<b>About This Guide .....</b>	<b>1</b>
Purpose .....	1
Audience .....	1
Typical User .....	1
Prerequisites .....	2
Topics Covered .....	2
Additional References .....	2
Text Conventions Used .....	3
Credits and Resources .....	4
Application Support Contacts .....	5
<b>Chapter 1</b>	
<b>About the Enterprise Vocabulary Services .....</b>	<b>7</b>
Introduction .....	7
Key EVS Terminologies .....	8
NCI Thesaurus (NCIT) .....	8
NCI Metathesaurus .....	8
Other EVS Terminology Resources .....	9
NCI Terminology Browser .....	9
FDA Terminology Data Standards .....	9
EVS Server Hosting .....	9
About the UMLS Metathesaurus .....	10
Preservation of Terminology .....	10
Disambiguation of Terminology .....	10
Defined Relationships .....	10
Categorization of Concepts .....	11
Related Online Resources .....	11
General .....	11
Technical .....	11
Other .....	12

## Chapter 2

### Description Logic and the NCI Thesaurus Semantic Model .....13

About Knowledge Representation .....	13
Frame-Based Representations .....	13
First Order Predicate Logic .....	14
Later Developments .....	15
About Description Logics and OWL .....	16
Concepts and Roles .....	18
<i>OWL Class Descriptions and Anonymous Classes</i> .....	18
<i>Subsumption and Disjointness Axioms</i> .....	21
<i>Equivalency and Necessary vs. Necessary and Sufficient Conditions</i> .....	21
<i>Property Restrictions and the Existential and Universal Qualifiers</i> .....	22
Description Logic in the NCI Thesaurus .....	23
OWL and the NCI Thesaurus .....	24
Identifiers .....	24
Class Expressions .....	25
Property Hierarchy .....	25
Property Inverses .....	25
Datatype Properties .....	25
Transitivity of Properties .....	25
More Precise Declaration of Property Domains and Ranges .....	25
Some vs. All Property Restrictions .....	25
Defining and Non-Defining Restrictions .....	25
Disease-Anatomy Relationships .....	25
Property Qualifiers .....	26
Associations .....	26
Exceptional Conditions .....	26

## Chapter 3

### Getting Started with Protégé .....27

Installing and Updating Protégé .....	27
Downloading the Installation Files .....	28
Preparing for the Installation .....	29
<i>Uninstalling Previous Versions of Protégé</i> .....	29
Installing the Main Application File .....	30
<i>Confirming that the Installation Directory was Created</i> .....	30
Installing Application Updates .....	31
Logging In to the Protégé Server .....	32
Using a Local Thesaurus Subset .....	35
Downloading the Subset File .....	35

Installing the Subset File .....	35
Opening the Local Subset File in Protégé .....	36
Setting Up a Local Project .....	38
<b>Chapter 4</b>	
<b>About the NCI Protégé Plug-In .....</b>	<b>41</b>
Overview .....	41
About the NCI Editor Tab .....	42
Class Browser .....	43
Edit Tab .....	45
<i>Basic Data Subtab</i> .....	46
<i>Relations Subtab</i> .....	47
<i>Properties Subtab</i> .....	48
Review Window .....	49
Split Tab .....	50
PreMerge Tab .....	51
Merge Tab .....	52
PreRetire Tab .....	53
Retire Tab .....	54
Report Writer Tab .....	55
Batch Loader tab .....	56
Batch Editor Tab .....	57
Partonomy Tree Tab .....	58
Copy Tab .....	59
About the Advanced Query Tab .....	60
About the Console .....	62
Setting the Console Display Capacity .....	63
Capturing Information for Troubleshooting .....	64
<i>Capturing a Thread Dump</i> .....	64
<i>Copying the Console Window Contents</i> .....	64
<i>Submitting a Bug Report and File Attachment to GForge</i> .....	65
<b>Chapter 5</b>	
<b>Searching: Simple and Advanced .....</b>	<b>67</b>
Simple vs. Advanced Searching .....	67
Performing Simple Searches .....	68
<i>Using the Simple Search Field and Button</i> .....	69
<i>Selecting and Reviewing Search Results</i> .....	70
Building Advanced Queries .....	71
<i>Advanced Query Syntax</i> .....	72
<i>Tips for Using the Query-Building Interface</i> .....	72

<i>Building a Simple Property Query: Preferred Name</i> .....	74
<i>Building a Complex Property Query: FULL_SYN</i> .....	75
<i>Building a Combined Property Query: Preferred_Name and FULL_SYN</i> .....	77
<i>Building a Restriction-Based Query</i> .....	81
Configuring the Advanced Query Tab .....	84
Changing the Default Search Property .....	85
Changing the Default Search Parameter .....	86
Verifying the New Search Configuration .....	87
Search Properties and Values in the protege.properties File .....	87
Available Data Types from SMI .....	88

## Chapter 6

<b>Using Basic Editing Features</b> .....	<b>89</b>
About Classes .....	89
Class Identifiers .....	90
Class Relationships .....	90
Class Properties .....	90
<i>About the FULL_SYN Property</i> .....	91
<i>About the Definition Property</i> .....	91
Reviewing Changes As You Work .....	93
Creating a Class .....	94
Treeing Classes .....	98
Adding a Parent Class .....	99
Modifying a Parent Class .....	101
Deleting a Parent Class .....	102
Asserting Annotation Properties .....	103
Tips for Working with Properties .....	103
Adding a Full Synonym .....	104
Modifying a Full Synonym .....	105
Deleting a Full Synonym .....	105
Modifying a Definition .....	106
Asserting Relations .....	107
Adding a Simple Restriction .....	107
Modifying a Restriction .....	111
Deleting a Restriction .....	114
Adding a Role Group .....	115
<i>Adding Multiple Restrictions</i> .....	115
<i>Building a Role Group Expression</i> .....	118
Adding an Association .....	119

## Chapter 7

### Using Advanced Editing Features .....123

Splitting a Class .....	124
When to Consider Splitting a Class .....	124
Steps for Splitting a Class .....	124
Merging Classes .....	127
When to Consider Merging Classes .....	127
Pre-Merge: Flagging Classes to be Merged .....	127
Merging Flagged Classes (Workflow Managers Only) .....	130
Retiring a Class .....	131
When to Consider Retiring a Class .....	131
Pre-Retire: Flagging a Class for Retirement .....	131
<i>Selecting a Class</i> .....	131
<i>Re-treeing the Subclasses</i> .....	132
<i>Completing the PreRetire Task</i> .....	135
Retiring a Flagged Class (Workflow Managers Only) .....	135
Using the Report Writer .....	136
(Optional) Creating an Output Directory and Files .....	136
Generating a Report .....	136
Loading a Batch of Classes for Editing .....	142
Editing a Batch of Classes .....	146
Generating a Partonomy Tree .....	150
Copying a Class .....	152

## Appendix A

### TDE vs. Protégé Terminology .....155

### Index .....157



# LIST OF FIGURES

Figure 2.1. An earthquake in a semantic network of news stories.....	14
Figure 2.2. “Thing” OWL class and some subtypes of the anatomic domain .....	19
Figure 2.3. Effect of complement constructor on anonymous classes described by part_of role with domain AnatomicPart and range Anatomy .....	20
Figure 2.4. Hierarchy of named classes A, B, C, D, E, with class C as defined .....	22
Figure 3.1. Add or Remove Programs window.....	29
Figure 3.2. Winzip Self-Extractor window - New Installation .....	30
Figure 3.3. Winzip Self-Extractor window - Package Update.....	31
Figure 3.4. Welcome to Protégé window .....	32
Figure 3.5. Open Project window - Server option.....	33
Figure 3.6. Select Project window.....	33
Figure 3.7. Main Protégé window .....	34
Figure 3.8. Welcome to Protégé window .....	36
Figure 3.9. Open Project window - File option.....	37
Figure 3.10. Protégé standard interface with no visible plug-ins.....	38
Figure 3.11. Configure file window with ordered tab widgets .....	39
Figure 3.12. New local configuration.....	39
Figure 4.1. Main Protégé window with NCI Editor tab and subtabs .....	42
Figure 4.2. Class Browser .....	43
Figure 4.3. Edit tab .....	45
Figure 4.4. Basic Data subtab.....	46
Figure 4.5. Relations subtab .....	47
Figure 4.6. Properties subtab .....	48
Figure 4.7. Review window .....	49
Figure 4.8. Split tab .....	50
Figure 4.9. PreMerge tab .....	51
Figure 4.10. Merge tab .....	52
Figure 4.11. PreRetire tab.....	53
Figure 4.12. Retire tab .....	54
Figure 4.13. Report Writer tab and Report Writer window .....	55
Figure 4.14. Batch Loader tab .....	56
Figure 4.15. Batch Editor tab .....	57
Figure 4.16. Partonomy Tree tab.....	58
Figure 4.17. Copy tab .....	59
Figure 4.18. Advanced Query tab .....	60
Figure 4.19. Console window.....	62

Figure 4.20. Console window menu - Properties command .....	63
Figure 4.21. Console Properties window .....	63
Figure 4.22. Command line window menu - Select All command .....	64
Figure 4.23. Tracker page - Submit New link .....	65
Figure 5.1. Simple Search field and button .....	67
Figure 5.2. NCI Editor tab with Simple Search field and button .....	68
Figure 5.3. Select Property window with Simple Search field and button .....	68
Figure 5.4. Simple Search field and Search for Class button .....	69
Figure 5.5. Results for simple search .....	69
Figure 5.6. Edit tab with Basic Data for a selected result .....	70
Figure 5.7. Advanced Query tab .....	71
Figure 5.8. Default query .....	72
Figure 5.9. Parameter list .....	73
Figure 5.10. Default query .....	74
Figure 5.11. Default query .....	75
Figure 5.12. Select Slot window .....	75
Figure 5.13. Parameter list - starts with .....	76
Figure 5.14. Finished query .....	76
Figure 5.15. Default query .....	77
Figure 5.16. Selecting from the Parameter list .....	77
Figure 5.17. First completed query .....	77
Figure 5.18. Select Slot window .....	78
Figure 5.19. Second completed query .....	78
Figure 5.20. Combined queries .....	79
Figure 5.21. Full query with search results .....	79
Figure 5.22. Removing the default query .....	81
Figure 5.23. Initial setup for restriction-based query .....	81
Figure 5.24. Select Property window .....	82
Figure 5.25. Restriction-based query with search property .....	82
Figure 5.26. Restriction-based query with completed fields .....	83
Figure 5.27. Search results for restriction-based query .....	83
Figure 5.28. Default query - Preferred Name with exact matching .....	84
Figure 5.29. Default query with new search property and search parameter .....	84
Figure 5.30. Configure file window with protege.properties tab selected .....	85
Figure 5.31. Configure file window - Property Files tab with editable field .....	85
Figure 5.32. Configure file window with new search property and parameter .....	86
Figure 6.1. Class information displayed on Edit tab - Basic Data subtab .....	90
Figure 6.2. Definitions and Qualifiers panels .....	91
Figure 6.3. Review window .....	93
Figure 6.4. Creating a new class .....	94
Figure 6.5. Create Subclass window .....	95
Figure 6.6. Select a superclass window .....	95
Figure 6.7. Advanced Query window - Search for Class .....	96
Figure 6.8. Create Subclass window with completed fields .....	96
Figure 6.9. New class information .....	97
Figure 6.10. Edit Tab - Parent Class panel .....	99
Figure 6.11. Select a class window - Defining check box .....	100

Figure 6.12. Parent Class panel with new class .....	100
Figure 6.13. Modify Named Superclass window .....	101
Figure 6.14. Select a named class window .....	101
Figure 6.15. Modified parent class .....	102
Figure 6.16. Create FULL_SYN Annotation Property window .....	104
Figure 6.17. New synonym .....	104
Figure 6.18. Edit FULL_SYN Annotation Property window .....	105
Figure 6.19. Edit DEFINITION Annotation Property window .....	106
Figure 6.20. Edit tab - Relations subtab - Restrictions panel .....	107
Figure 6.21. Create a Restriction window (first of two windows) .....	108
Figure 6.22. Create a Restriction window (second of two windows) .....	108
Figure 6.23. Select a named class window .....	109
Figure 6.24. Create a Restriction window with Filler value .....	109
Figure 6.25. Newly added restriction .....	110
Figure 6.26. Inherited restriction .....	111
Figure 6.27. Edit a Restriction window .....	111
Figure 6.28. Modify a Restriction window .....	112
Figure 6.29. Select a named class window .....	112
Figure 6.30. Modify a Restriction window with new Filler value .....	113
Figure 6.31. Newly modified restriction .....	113
Figure 6.32. Edit tab - Relations subtab - Restrictions panel .....	115
Figure 6.33. Create a Restriction window (first of two windows) .....	115
Figure 6.34. Create a Restriction window (second of two windows) .....	116
Figure 6.35. Select a named class window .....	116
Figure 6.36. Create a Restriction window with Filler value .....	117
Figure 6.37. Multiple restrictions .....	117
Figure 6.38. Role group example .....	118
Figure 6.39. Role group expression .....	118
Figure 6.40. Relations subtab for Phagocytosis .....	119
Figure 6.41. Add an Object-Valued Property window .....	120
Figure 6.42. Select a Property window .....	120
Figure 6.43. Select a property value added .....	121
Figure 6.44. Add an Object-Valued Property window with stored values .....	121
Figure 6.45. Associations panel .....	121
Figure 7.1. Existing Concept pane .....	124
Figure 7.2. Enter Class Identifiers window .....	125
Figure 7.3. Split classes .....	125
Figure 7.4. Right-click shortcut menu for modifying split classes .....	126
Figure 7.5. Properties of new class .....	126
Figure 7.6. PreMerge tab with surviving and retiring concepts .....	128
Figure 7.7. Right-click shortcut menu for modifying pre-merged classes .....	128
Figure 7.8. Enter Notes window .....	129
Figure 7.9. Merging two classes .....	130
Figure 7.10. PreRetire tab with subclasses .....	131
Figure 7.11. Hiding the Class Browser .....	132
Figure 7.12. Selected subclass with Basic Data subtab displayed .....	132
Figure 7.13. Relations tab with one parent class .....	133

Figure 7.14. Select a class window .....	133
Figure 7.15. Buttons sets for inner, nested subtabs and outer tab .....	134
Figure 7.16. Enter Notes window .....	135
Figure 7.17. Report Writer tab - Generate button .....	137
Figure 7.18. Report Writer window for specifying an output file.....	137
Figure 7.19. File browser window .....	138
Figure 7.20. Level numbers for Exocrine Gland Fluid or Secretion.....	139
Figure 7.21. Report Writer window with completed values .....	139
Figure 7.22. Report Writer status window as it initially appears .....	140
Figure 7.23. Report with classes only - no attributes .....	140
Figure 7.24. Report with classes, subclasses, properties, and restrictions .....	141
Figure 7.25. Tab-delimited input file for batch load .....	142
Figure 7.26. Batch Loader tab .....	143
Figure 7.27. Batch Loader window .....	143
Figure 7.28. Browse window for batch load files .....	144
Figure 7.29. Batch Loader window with completed file paths and names .....	144
Figure 7.30. Newly imported batch load.....	145
Figure 7.31. Tab-delimited input file for batch load .....	147
Figure 7.32. Batch Editor tab .....	147
Figure 7.33. Batch Editor window .....	148
Figure 7.34. Browse window for batch edit files .....	148
Figure 7.35. Batch Editor window with completed file paths and names.....	149
Figure 7.36. Newly imported batch edit.....	149
Figure 7.37. Partonomy Tree tab.....	150
Figure 7.38. Select Transitive Properties window .....	150
Figure 7.39. Partonomy tree for Blood class.....	151
Figure 7.40. Copy tab with original class.....	152
Figure 7.41. Enter Class Identifiers window .....	153
Figure 7.42. Copy tab with newly created class.....	153
Figure 7.43. Right-click menu for modifying original and cloned classes .....	154

# ABOUT THIS GUIDE

This section introduces you to the *NCI Editor's Guide to Using Protégé*. It includes the following topics:

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

## Purpose

---

This guide is a comprehensive reference for editors using the NCI Protégé Plug-in. It discusses the semantic model that supports the Protégé application. It also explains how to install Protégé and how to use the NCI Protégé Plug-in to build and manage ontologies for the NCI Thesaurus.

## Audience

---

### Typical User

This guide is designed for the following users:

- Existing NCI ontology editors who have used the TDE application and are now using Protégé
- New NCI ontology editors who need background information on the semantic model used at the NCI, an understanding of the editors' workflow, and procedural information for using 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 the NCI Protégé Plug-in, see *Additional References* on this page.

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

- *Chapter 1* is an overview of the Enterprise Vocabulary Services (EVS).
- *Chapter 2* is 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 Protégé installation files, how to install the application, how to log in, and how to set up a local subset of the thesaurus for learning the interface.
- *Chapter 4* provides an overview of the NCI Protégé Plug-In interface.
- *Chapter 5* explains simple and advanced searching techniques and provides several query examples.
- *Chapter 6* explains how to use the NCI Protégé Plug-In to perform core editing tasks.
- *Chapter 7* explains how to perform more advanced tasks such as splitting, pre-merging, and pre-retiring classes.
- *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 the NCI Protégé Plug-in, 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](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.

<b>Convention</b>	<b>Description</b>	<b>Example</b>
<b>Bold</b>	Highlights names of option buttons, check boxes, drop-down menus, menu commands, command buttons, or icons.	Click <b>Search</b> .
<u>URL</u>	Indicates a Web address.	<a href="http://domain.com">http://domain.com</a>
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.
<i>Italics</i>	Highlights references to other documents, sections, figures, and tables.	See <i>Figure 4.5</i> .
<b><i>Italic boldface monospace type</i></b>	Represents text that you type.	In the <b>New Subset</b> text box, enter <b><i>Proprietary Proteins</i></b> .
<b>Note:</b>	Highlights information of particular importance	<b>Note:</b> 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.

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

## Application Support Contacts

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

<b>Support Option</b>	<b>Additional Information</b>
<b>For NCI Editors</b>	
GForge	If you are an NCI editor, post all bug reports and requests for enhancements on GForge, using the following URL: <a href="https://gforge.nci.nih.gov/tracker/?group_id=174">https://gforge.nci.nih.gov/tracker/?group_id=174</a> For instructions on reporting bugs, see <i>Submitting a Bug Report and File Attachment to GForge</i> on page 65.
<b>For General Users</b>	
E-mail	Write to <a href="mailto:ncicb@pop.nci.nih.gov">ncicb@pop.nci.nih.gov</a> . Include the following information: <ul style="list-style-type: none"> <li>• Your contact information, including your phone number</li> <li>• The name of the application that you are using</li> <li>• The URL (for Web-based applications)</li> <li>• A description of the problem and the steps required to recreate it</li> <li>• The text of any error messages you have received.</li> </ul>
Web	NCICB support: <a href="http://ncicb.nci.nih.gov/NCICB/support">http://ncicb.nci.nih.gov/NCICB/support</a> NCI Protégé online support forums: <a href="https://gforge.nci.nih.gov/forum/?group_id=174">https://gforge.nci.nih.gov/forum/?group_id=174</a>
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), its terminologies, and related resources. It includes the following topics:

- *Introduction* on this page
- *Key EVS Terminologies* on page 8
- *Other EVS Terminology Resources* on page 9
- *EVS Server Hosting* on page 9
- *About the UMLS Metathesaurus* on page 10
- *Related Online Resources* on page 11

### 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

### NCI Thesaurus (NCIT)

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

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

### NCI Metathesaurus

The NCI Metathesaurus is 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. It excludes certain proprietary vocabularies and includes others with restricted use.

Some of the NCI Metathesaurus vocabularies were developed locally by the NCI, and others were licensed. [Table 1.1](#) describes those vocabularies that were developed locally. Note that a limited model of the NCI Thesaurus is accessible through the Metathesaurus as the NCI Source.

<b>Vocabulary</b>	<b>Content</b>	<b>Usage</b>
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.
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*

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.

For more information about the UMLS Metathesaurus, see [About the UMLS Metathesaurus](#) on page 10.

## 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 Health Level Seven (HL7) and adopted by the FDA as a mechanism for exchanging information about medications.

## EVS Server Hosting

---

The caCORE EVS API provides access to the NCI Metaphrase, which hosts the Metathesaurus database, and 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.

## About the UMLS Metathesaurus

The NCI Metathesaurus is based on the National Library of Medicine's (NLM) Unified Medical Language System Metathesaurus (UMLS Metathesaurus). This section provides a brief overview of the UMLS Metathesaurus features that are relevant to accessing the NCI Metathesaurus. More detailed information about the UMLS Metathesaurus is available on the UMLS Knowledge Sources web site at <http://www.nlm.nih.gov/research/umls/umlsdoc.html>.

### Preservation of Terminology

The UMLS Metathesaurus is a unifying database of concepts that brings together terms occurring in over 100 different controlled vocabularies used in biomedicine. When editors add terms to the UMLS Metathesaurus, they preserve all of the original meanings, attributes, and relationships defined in the source vocabularies, and they retain explicit source information. They also add basic information about each concept and introduce new associations that help to establish synonymy and other relationships among concepts from different sources.

### Disambiguation of Terminology

Given the large number of related vocabularies incorporated in the UMLS Metathesaurus, instances occur in which the same concept may be known by many different names. In other cases, the same names are intended to convey different concepts. To avoid ambiguity, the UMLS uses an elaborate indexing system that assigns a *concept unique identifier* (CUI) to each concept name. Similarly, each unique concept name or string in the Metathesaurus is assigned a *string unique identifier* (SUI).

In cases where one string is associated with multiple concepts, a numeric tag is appended to that string to render it unique and to reflect its multiplicity. UMLS Metathesaurus editors may also create an alternative name for the concept that is more indicative of its intended interpretation. In such cases, all three concept names are preserved.

### Defined Relationships

Several types of relationships are defined in the UMLS Metathesaurus. The NCI Metaphrase interface captures the four relationships described in [Table 1.2](#).

<b>Relationship</b>	<b>Description</b>
Broader (RB)	The related concept has a more general meaning.
Narrower (RN)	The related concept has a more specific meaning.
Synonym (SY)	The two concepts are synonymous.
Other related (RO)	The relationship is not specified, but it is something other than broader, narrower, or synonymous.

*Table 1.2 Relationships defined in the UMLS Thesaurus*

## Categorization of Concepts

The UMLS *Semantic Network* is an independent construct that provides consistent categorization for all concepts contained in the UMLS Metathesaurus and defines a useful set of relationships among those concepts. As of the 2005AC release, the Semantic Network defined a set of 135 basic semantic types or categories that could be assigned to concepts, as well as 54 relationships that could hold among these types.

Major groupings of semantic types include organisms, anatomical structures, biologic function, chemicals, events, physical objects, and concepts or ideas. Each UMLS Metathesaurus concept is assigned at least one semantic type. In some cases, several concepts are assigned. In all cases, the most specific semantic type available in the network hierarchy is assigned to the concept.

## Related Online Resources

### General

*Table 1.3* lists general EVS resources that are available online.

<b>Resource</b>	<b>Location</b>
EVS main Web site	<a href="http://evs.nci.nih.gov">http://evs.nci.nih.gov</a>
NCI Thesaurus (NCIT) and NCI Terminology Browser	<a href="http://nciterms.nci.nih.gov/NCIBrowser/Dictionary.do">http://nciterms.nci.nih.gov/NCIBrowser/Dictionary.do</a>
NCI Metathesaurus	<a href="http://ncimeta.nci.nih.gov/MetaServlet/">http://ncimeta.nci.nih.gov/MetaServlet/</a>
NCI Distributed Terminology Server	<a href="http://nciterms.nci.nih.gov/NCIBrowser/Dictionary.do">http://nciterms.nci.nih.gov/NCIBrowser/Dictionary.do</a>

*Table 1.3 EVS online resources (general)*

### Technical

*Table 1.4* lists online technical resources that are related to the EVS infrastructure and caCORE API.

<b>Resource</b>	<b>Location</b>
caCORE Technical Guide	<a href="http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview">http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview</a>
caCORE API	<a href="http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/caBIO">http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/caBIO</a>
EVS Technical Overview	<a href="http://evs.nci.nih.gov/moreInformation/">http://evs.nci.nih.gov/moreInformation/</a>
EVS Production and Server Environment	<a href="http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary/implementation">http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary/implementation</a>

*Table 1.4 EVS online resources (technical)*

## Other

*Table 1.5* describes online resources that are related to other topics in this chapter.

<b>Resource</b>	<b>Location</b>
FDA Terminology Data Standards	<a href="http://www.fda.gov/oc/datacouncil/spl.html">http://www.fda.gov/oc/datacouncil/spl.html</a>
UMLS Metathesaurus	<a href="http://www.nlm.nih.gov/research/umls/umlsdoc.html">http://www.nlm.nih.gov/research/umls/umlsdoc.html</a>

*Table 1.5 Other online resources*

## CHAPTER 2

# DESCRIPTION LOGIC AND THE NCI THESAURUS SEMANTIC MODEL

This chapter provides an overview of knowledge representations, description logic, and the use of description logic in the NCI Thesaurus. It includes the following topics:

- *About Knowledge Representation* on this page
- *About Description Logics and OWL* on page 16
- *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 utilize *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.<sup>1</sup>

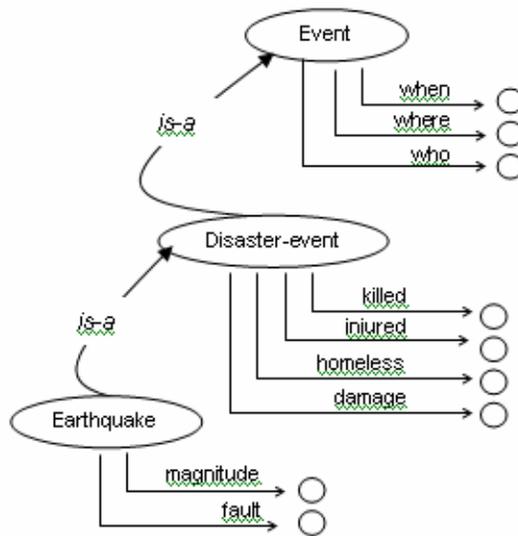


Figure 2.1 An earthquake in a semantic network of news stories

### 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.

<b>Propositional Logic</b>	<b>First-Order Predicate Logic</b>
All men are mortal. Socrates is a man. Socrates is mortal.	$\forall x : \text{Man}(x) \Rightarrow \text{Mortal}(x)$ $\text{Man}(\text{Socrates}) \Rightarrow \text{Mortal}(\text{Socrates})$

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  $\rightarrow$  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 in order 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.

## Later Developments

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. (See [Table 2.2](#).) The difference among DLs boils down to the construction operations (see below) 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](#) shows various types of DL expressivities.<sup>2</sup>

<b>Abbreviation</b>	<b>Description</b>
$\mathcal{AL}$	Attributive language. This is the base language which allows atomic negation, concept intersection, universal restrictions, and limited existential quantification.
$\mathcal{FL}$ -	A sub-language of $\mathcal{AL}$ , which is obtained by disallowing atomic negation.
$\mathcal{FL}_0$	A sub-language of $\mathcal{FL}$ -, which is obtained by disallowing 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.

*Table 2.2 DL expressivities*

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

<b>Abbreviation</b>	<b>Description</b>
$\mathcal{E}$	Full existential qualification (Existential restrictions that have fillers other than owl:thing).
$\mathcal{U}$	Concept union.
$(\mathcal{D})$	Use of datatype properties, data values or data types.

Table 2.2 DL expressivities (Continued)

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

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 *SHOIN(D)* DL. (See Table 2.2.) Although the NCI Thesaurus is built as an OWL DL (originally as an *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. The expression syntax used in OWL is shown in *Table 2.3* on page 24.

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 intentional 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 the NCI Protégé Plug-In; however, some terms might show vestiges of the previous editing environment during a transition period.

Be aware of these differences in terminology; 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 last 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](#). 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 ( $\exists$ ) represents the existential qualifier.

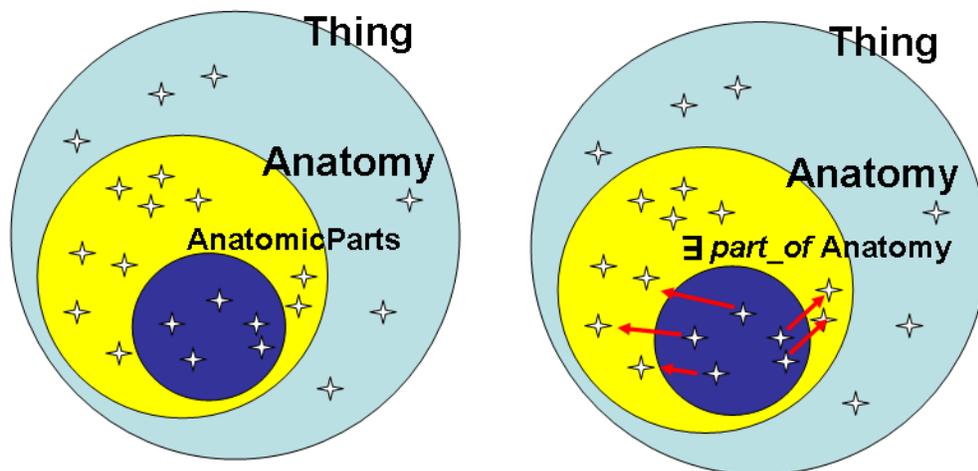


Figure 2.2 "Thing" OWL class and some subtypes of the anatomic domain

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 the NCI Protégé plug-in, intersections are not directly supported; intersections are used, however, in class expressions termed *role groups* (discussed below), but the plug-in does not make the details of the construction explicit to the editors.

As with the intersection, the NCI plug-in 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 the NCI Protégé Plug-In, 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](#).

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.

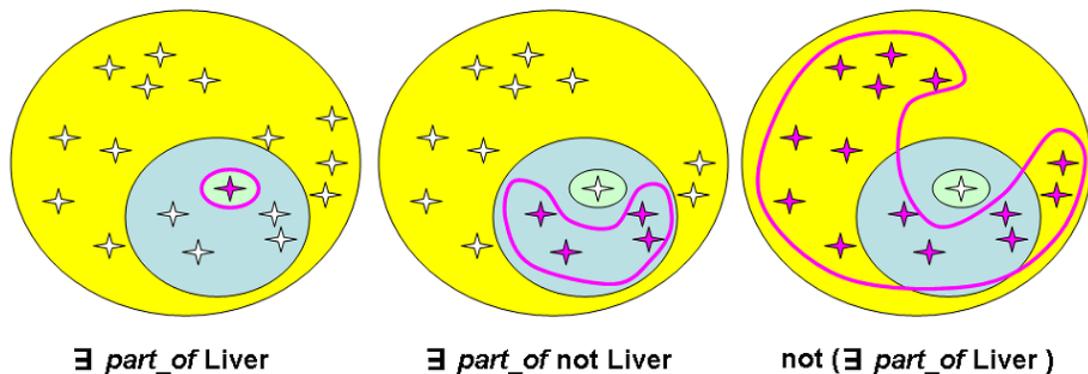


Figure 2.3 Effect of complement constructor on anonymous classes described by *part\_of* role with domain *AnatomicPart* and range *Anatomy*

For reference, in [Figure 2.3](#) the outermost yellow circle represents the *Anatomy* class, the light blue middle circle represents the class *AnatomicPart*, 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 *AnatomicPart* 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, the NCI Protégé Plug-In 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.

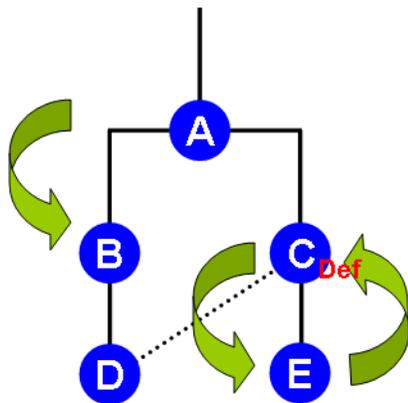


Figure 2.4 Hierarchy of named classes A, B, C, D, E, with class C as defined

### 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 ( $\exists$ ), 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 ( $\forall$ ). 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. The NCI Protégé Plug-In 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 the NCI Protégé plug-in, a tab plug-in that extends Protégé to support the NCI-specific editing environment. This plug-in provides editing capabilities that are not available in the Protégé OWL plug-in.

To facilitate use of the NCI Protégé plug-in, 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.<sup>3</sup>

OWL Element	Symbol	Key	Example
allValuesFrom	$\forall$	*	$\forall$ children Male
someValuesFrom	$\exists$	?	$\exists$ children Lawyer
hasValue	$\sqcap$	\$	rich $\sqcap$ true
cardinality	=	=	children = 3
minCardinality	$\geq$	>	children $\geq$ 3
maxCardinality	$\leq$	<	children $\leq$ 3
complementOf	$\neg$	!	$\neg$ Parent
intersectionOf	$\cap$	&	Human $\cap$ Male
unionOf	$\cup$		Doctor $\cup$ Lawyer
enumeration	{...}	{ }	{male female}

Table 2.3 Owl expression syntax used in Protégé

**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. The NCI Protégé Plug-In 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.

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>).

## 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 *Gastrointestinal System* or any of its parts:

*Gastrointestinal Carcinoma Disease\_Has\_Associated\_Anatomic\_Site some (Gastrointestinal\_System or (some Anatomic\_Structure\_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 how to download and install the Protégé application files, how to update your current installation, how to log in to the application, and how to install a local subset file that you can use to learn the Protégé interface. It includes the following topics:

- *Installing and Updating Protégé* on this page
- *Installing Application Updates* on page 31
- *Logging In to the Protégé Server* on page 32
- *Using a Local Thesaurus Subset* on page 35

### Installing and Updating Protégé

---

Protégé installation files, update files, and other related files are available for download on GForge. They are stored in the Docs area of the NCICB EVS Collaborative Terminology Development Tools project. The URL for the project is <http://gforge.nci.nih.gov/projects/protégegui/>.

Whenever the latest packaged updates and release notes are available for Protégé, you will receive an e-mail announcement with a link to the file location on GForge. When you download the package file, also download and read the release notes.

---

**Note:** File and folder names in this chapter reflect the current version of Protégé as of this writing. If you are installing a newer version, the names will include the current version number.

---

## Downloading the Installation Files

To download the installation files for Protégé and the NCI plug-ins, follow these steps:

1. Access the GForge site for the NCI Protégé GUI project at <http://gforge.nci.nih.gov/projects/protegegui/>.
2. Click the **Files** tab.
3. Click the name of one of the files listed in *Table 3.1*.
4. Save the file to a temporary directory (such as *C:\temp* or *C:\Downloads*) on your hard drive.

<b>Category</b>	<b>File name</b>	<b>Purpose</b>
Base Installation	<i>ProtegeClient-1.1.3.exe</i> (Windows users) <i>ProtegeClient-1.1.3.zip</i> (Mac users)	Installs the Protégé application. <b>Note:</b> The steps in this guide are geared to Windows users.
Documentation	<i>Beta_1_ReleaseNotes.doc</i>	Provides an overview of new features, known bugs, limitations, and workarounds in the current release; a list of all bugs fixed in the current release; and any documented information written after manuals were released.
OWL File (Optional)	<i>Thesaurus-ByName-060926-Retired.zip</i>	Provides a subset of the NCI Thesaurus that you can install on your hard drive and use as a practice file for learning the Protégé interface.  For more information, see <i>Using a Local Thesaurus Subset</i> on page 35.
Package	No current package releases as of this writing.	Extracts and installs all of the latest files needed to update your current Protégé installation.

*Table 3.1 Installation files for Protégé and NCI-specific plug-ins*

5. Note the name of the directory in which you stored the files.

---

**Note:** The steps in this guide are written for Windows users and assume that you are using WinZip. If you are a Mac user, consult the operating system Help for information about installing applications from zip files.

---

## Preparing for the Installation

### Uninstalling Previous Versions of Protégé

Before installing a new version of Protégé, remove the current version that is installed on your computer. Older beta versions may appear in the Currently Installed Programs list of the Windows Control Panel. Later versions do not appear in the list and can be deleted directly from Windows Explorer.

**Note:** Uninstalling Protégé and reinstalling a new version is different from updating the application. If your Protégé administrator instructs you to download and execute a package file to update your current Protégé installation, see *Installing Application Updates* on page 31. Otherwise, follow the steps in this section.

When uninstalling Protégé, follow the steps detailed in the next two sections:

- *Uninstalling a Version That Appears in the Control Panel List* on this page
- *Uninstalling Versions That Do Not Appear in the Control Panel List* on page 30

#### *Uninstalling a Version That Appears in the Control Panel List*

First, check to see if Protégé appears in the Windows Control Panel list. If you see it, follow these steps to uninstall it:

1. On the Windows Desktop, follow this path:  
**Start > Control Panel > Add or Remove Programs**
2. In the Add or Remove Programs window, locate the entry for Protégé, as shown in *Figure 3.1*.
3. If you do not see Protégé in the list of currently installed programs, skip to *Uninstalling Versions That Do Not Appear in the Control Panel List* on page 30. Otherwise, select the entry, then click the **Change/Remove** button.

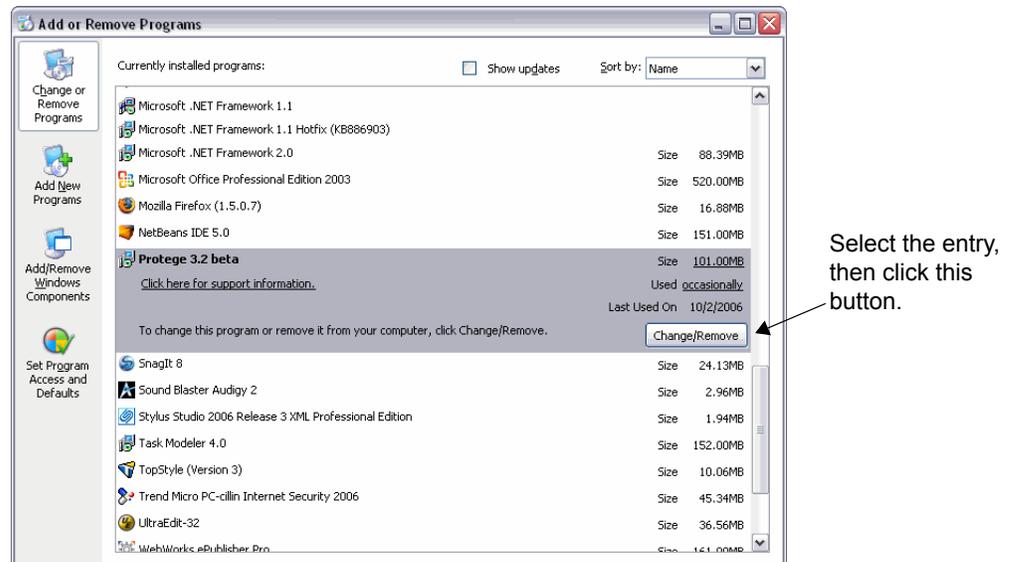


Figure 3.1 Add or Remove Programs window

Once the program is removed, the entry disappears from the list.

### *Uninstalling Versions That Do Not Appear in the Control Panel List*

To uninstall a version of Protégé that does not appear in the Control Panel list, follow these steps:

1. Using Windows Explorer, locate the following directory:

*C:\Program Files\Protege.Client-1.1.3*

**Note:** The name of the bottom-level folder shows the current version number for Protégé.

2. Delete the entire directory.

---

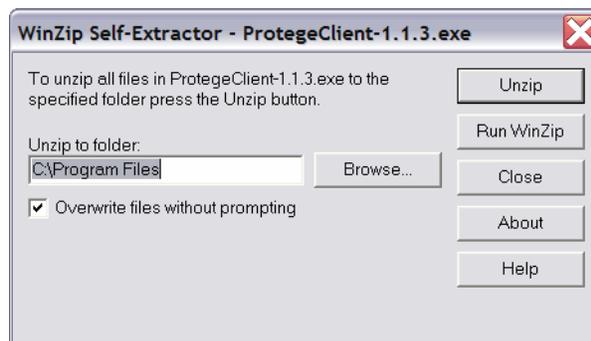
**Note:** Based on testing results by the Protégé Development team, you do not need to delete related Windows Registry entries.

---

### Installing the Main Application File

To install the main Protégé application file (e.g., *ProtegeClient-1.1.3.exe*), follow these steps:

1. Download the application file as specified in [Downloading the Installation Files](#) on page 28.
2. Locate the *ProtegeClient-1.1.3.exe* file (or file with current version name) on your hard drive, then double-click the file name. The WinZip Self-Extractor window opens, as shown in [Figure 3.2](#).



*Figure 3.2 Winzip Self-Extractor window - New Installation*

3. Ensure that the **Overwrite files without prompting** box is checked, then click **Unzip**.
4. When the WinZip Self-Extractor message window appears, click **OK** to close the message window.
5. Click **Close** to close the WinZip Self-Extractor window.

### Confirming that the Installation Directory was Created

Using Windows Explorer, browse to confirm that the following directory was created on your hard drive:

*C:\Program Files\Protege.Client-1.1.3*

---

**Note:** The name of the bottom-level folder shows the current version number for Protégé.

---

## Installing Application Updates

The Protégé development team periodically issues new versions of Java Archive (*jar*) files, as well as other files used to update the Protégé configuration. These files are used for such purposes as fixing bugs, implementing enhancements, and tuning application performance. The latest files are posted on GForge in a self-extracting, compressed *package* file.

**Note:** Whenever the latest packaged updates and release notes are available for Protégé, you will receive an e-mail announcement with a link to the file location on GForge. To download the file, follow the steps in [Downloading the Installation Files](#) on page 28, and download the files listed under the *Documentation* and *Package* categories.

To run the package file and install the updates, follow these steps:

1. Download the application file as specified in [Downloading the Installation Files](#) on page 28.
2. Locate the current package file (name varies) on your hard drive, then double-click the file name. The WinZip Self-Extractor window opens, as shown in [Figure 3.3](#).

**Note:** Unless you have installed Protégé to a custom directory (not recommended), leave the specified directory path set to *C:\Program Files*. The extractor will automatically write the files to the current Protégé application folder.

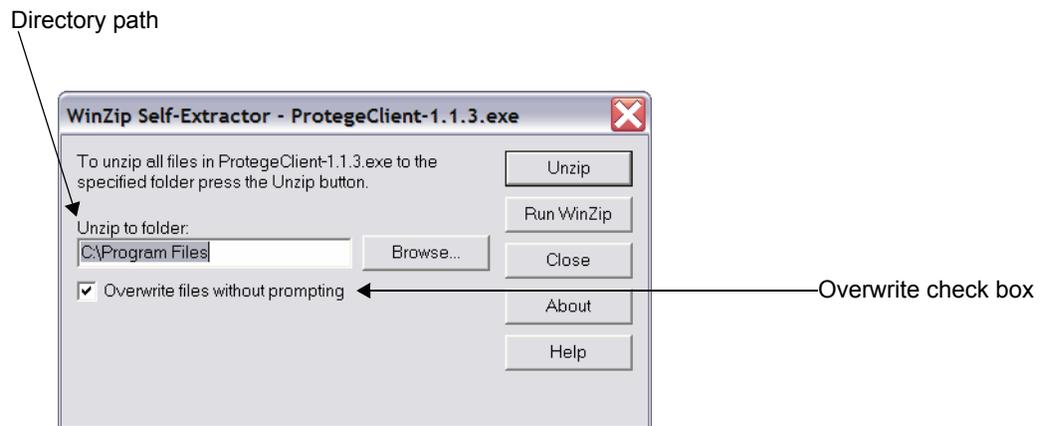


Figure 3.3 Winzip Self-Extractor window - Package Update

3. Ensure that the **Overwrite files without prompting** box is checked, then click **Unzip**.
4. When the WinZip Self-Extractor message window appears, click **OK** to close the message window.
5. Click **Close** to close the WinZip Self-Extractor window.

## Logging In to the Protégé Server

**Note:** If you do not already have a user name and password for logging in to Protégé, contact your local EVS administrator.

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

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

*C:\Program Files\Protege.Client-1.1.3*

**Note:** The name of the bottom-level folder shows the current version of Protégé.

2. Double-click the following file: *run\_protege.bat*.

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

- The Protégé 3.2 application window, and
- The Welcome to Protégé window, displayed in front.

3. In the Welcome to Protégé window, click the **Open Existing File** button, as shown in *Figure 3.4*.

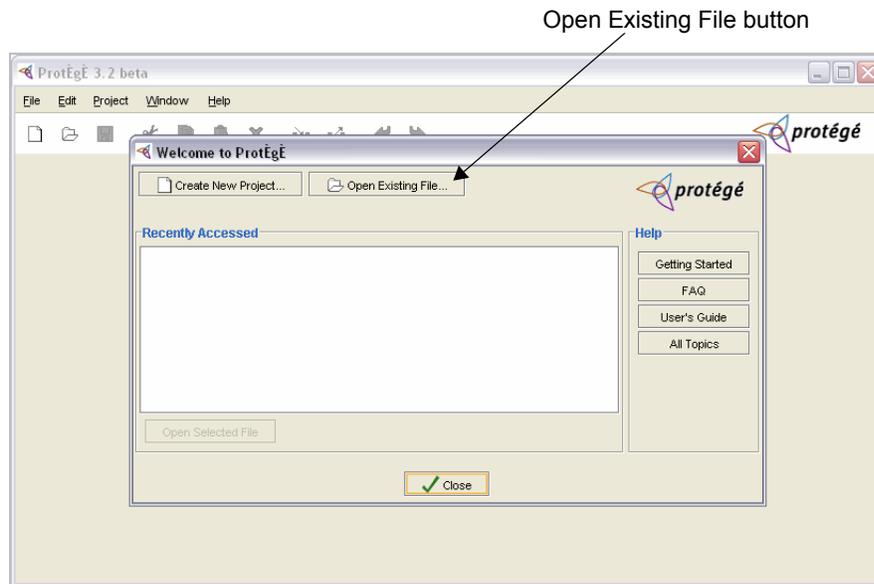


Figure 3.4 Welcome to Protégé window

- In the Open Project window, click the **Server** button on the lower left.
- Enter the **Host Machine Name**. Your Protégé administrator will provide this information.

**Note:** After the first time you log in, the **Host Machine Name** field should automatically display the host machine name.

The Open Project window should now resemble [Figure 3.5](#).

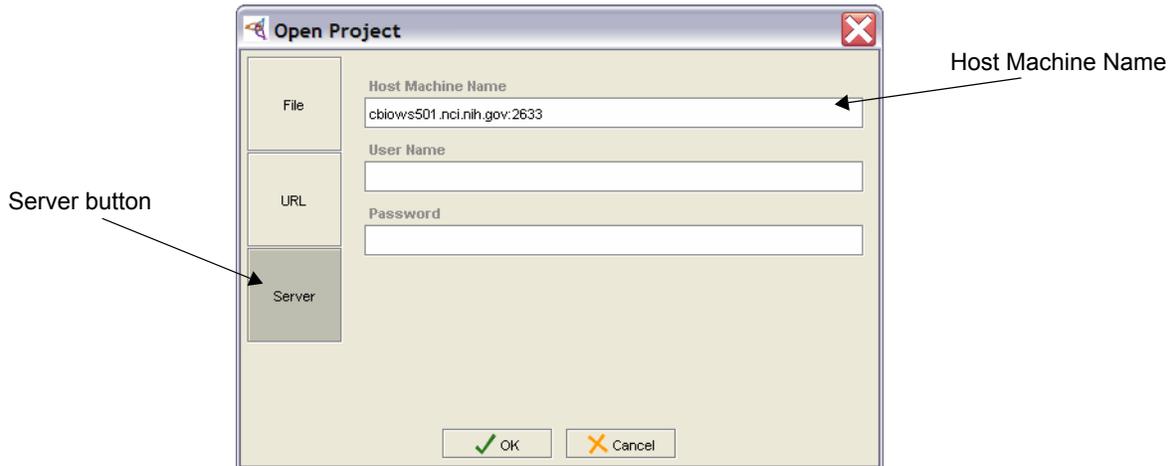


Figure 3.5 Open Project window - Server option

- Enter your user name and password, then click **OK**. The Select Project window opens, as shown in [Figure 3.6](#).
- Select the current project version (for example, **NCIThesaurus**), then click **OK**.

**Note:** The project number may differ from the one shown in [Figure 3.6](#). Your Protégé administrator will give you the current project name.

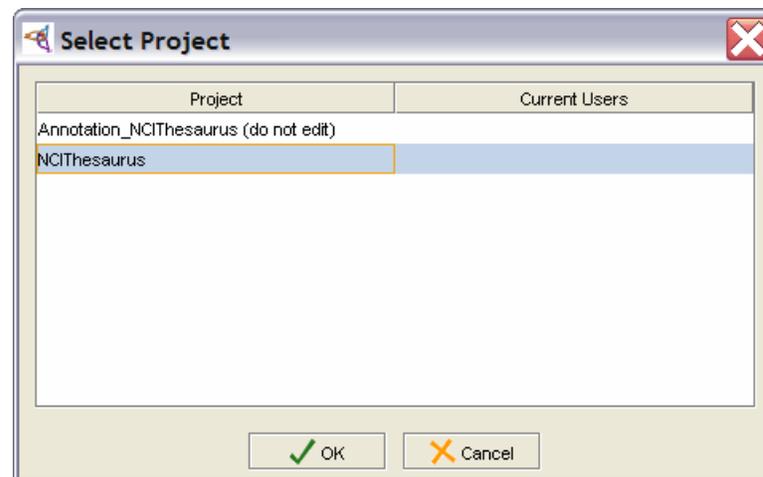


Figure 3.6 Select Project window

Protégé opens with the **NCI Editor** tab enabled, as shown in *Figure 3.7*.

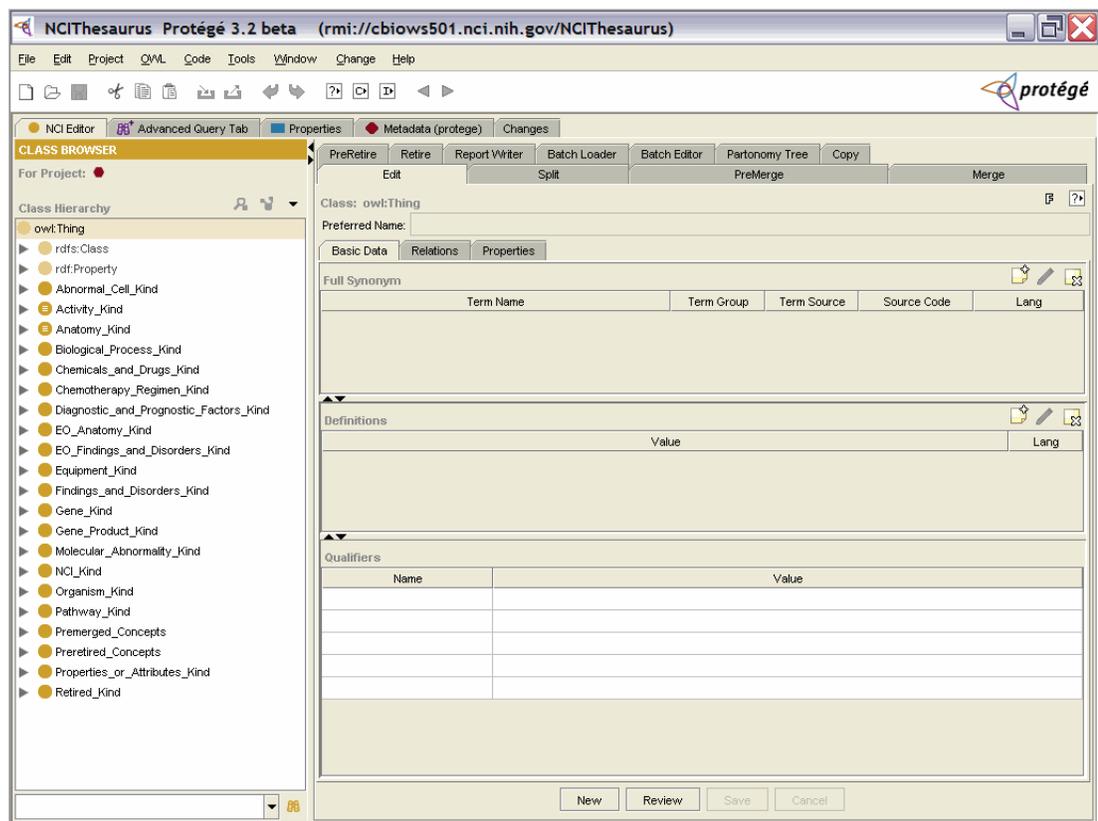


Figure 3.7 Main Protégé window

## Using a Local Thesaurus Subset

As an NCI editor, you normally work with Protégé while connected to a server. If you want to practice using 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 Protégé, the NCI plug-in tabs are not displayed. You will need to change the local tab configuration to match the tab display used on the Protégé server. Once you set up the tabs and tab order (explained in this section), 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 Protégé. You can select it whenever you want to run Protégé locally.

---

**Note:** Working with a local project will not affect your ability to log in to the server and do your regular work. Protégé enables you to freely alternate between your local file and the server database.

---

### Downloading the Subset File

To obtain the subset file and install it on your hard drive, follow these steps:

1. Navigate to the following location on GForge:  
<https://gforge.nci.nih.gov/projects/protegegui/>
2. Click the **Files** tab.
3. Under the *OWL File* category, select the file named **Thesaurus-ByName-060926-Retired.zip**.
4. Save the file to a directory on your hard drive, noting where you saved it.

### Installing the Subset File

To install the subset file, follow these steps:

1. Navigate to the directory where you downloaded the file.
2. (Optional) If you plan to store the file in a different location from the download directory, copy the file and paste it to the new location.  
**Note:** Do not copy the test file to the current Protégé installation folder. Instead, create a folder for local Protégé files and projects, and paste the file there. This will prevent you from inadvertently deleting Protégé application files.
3. Assuming that you are using WinZip, right-click the filename, then select **WinZip > Extract to here**. If you are using a different program, follow the program instructions for extracting compressed files.

The test file now appears in the folder list. The file name is identical to the compressed file, but with an added *.owl* extension.

## Opening the Local Subset File in Protégé

To open Protégé and direct it to use the local subset file, follow these steps:

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

*C:\Program Files\Protege.Client-1.1.3*

**Note:** The name of the folder shows the current version of Protégé.

The following windows open:

- The Protégé 3.2 beta window
  - The Welcome to Protégé window, displayed in front.
2. In the Welcome to Protégé window, click the **Open Existing File** button, as shown in [Figure 3.8](#).

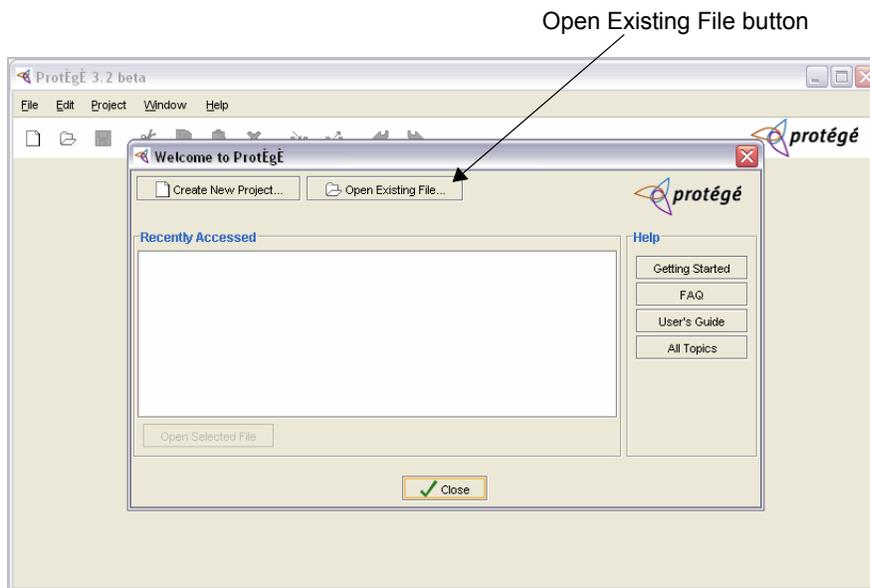


Figure 3.8 Welcome to Protégé window

3. In the Open Project window, click the **File** button in the upper left, as shown in [Figure 3.9](#).
4. Use the **Look In** list to browse for the file.

Click here...

...then browse to the file.

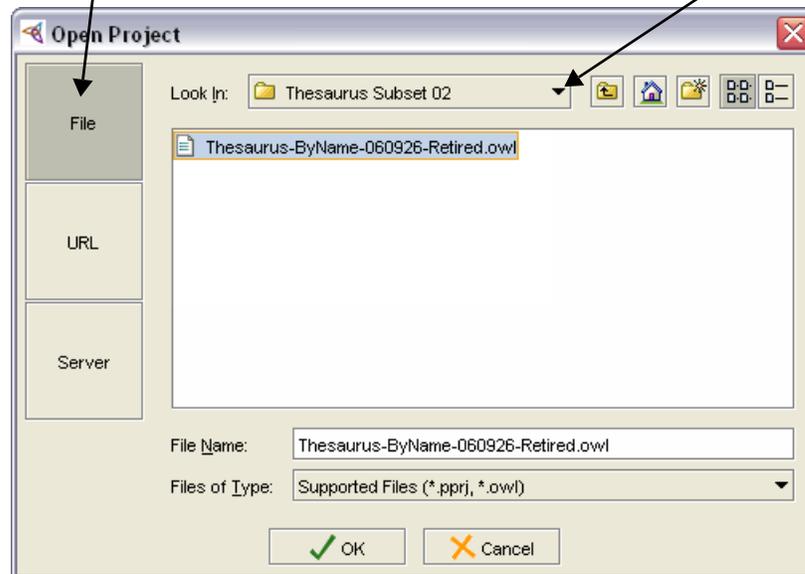


Figure 3.9 Open Project window - File option

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

Protégé opens, but the NCI plug-in tabs are not yet visible. In the next section, you will configure the workspace to mirror the server configuration.

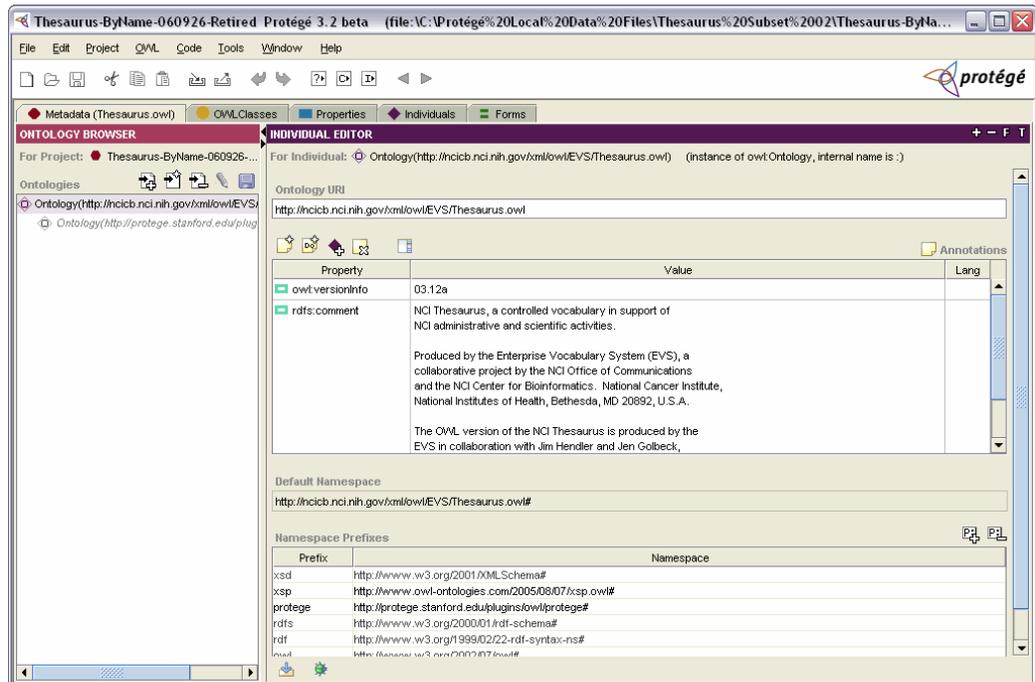


Figure 3.10 Protégé standard interface with no visible plug-ins

## Setting Up a Local Project

To change the Protégé tab configuration, 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 on the left, uncheck the boxes preceding each checked tab widget.
4. Check the box to the left of the **NCIEditTab**.
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:
  - **AdvancedQueryPlugin** (second position)
  - **OWLPropertiesTab** (third position)
  - **OWLMetadataTab** (fourth position)
  - **ChangesTab** (fifth position)

The Configure file window should now resemble [Figure 3.11](#).

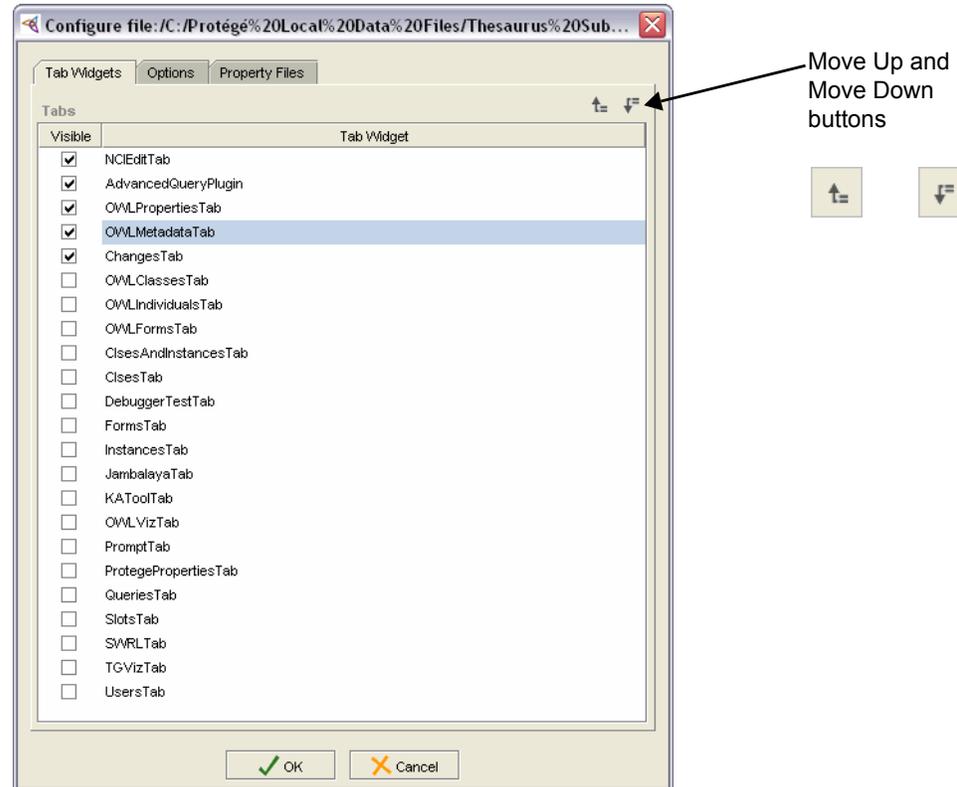


Figure 3.11 Configure file window with ordered tab widgets

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

As shown in [Figure 3.12](#), the tab display now matches the server configuration.

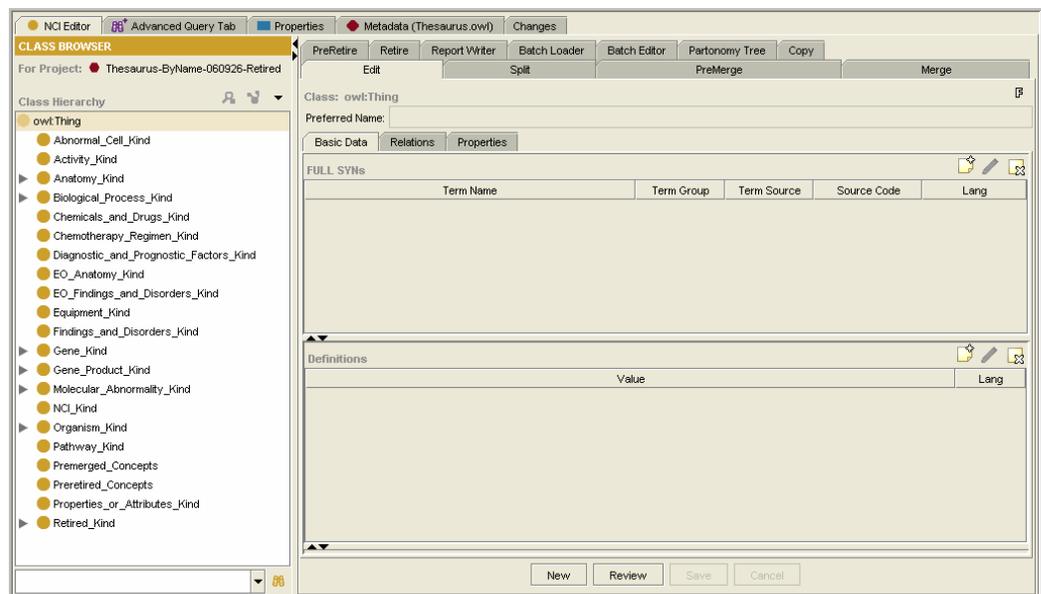


Figure 3.12 New local configuration

8. To save the new project, select either of the following menu commands:
  - To save a project with a name that matches the source file name, select **File > Save Project**.
  - To save a project with a new name, select **File > Save Project As**.

The next time you launch Protégé to run locally, your saved project will appear in the list of files.

# CHAPTER 4

## ABOUT THE NCI PROTÉGÉ PLUG-IN

This chapter provides an overview of the NCI Protégé Plug-In tabbed interface. It includes the following topics:

- *Overview* on this page
- *About the NCI Editor Tab* on page 42
- *About the Advanced Query Tab* on page 60
- *About the Console* on page 62

### Overview

---

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.

The NCI Protégé Plug-In is a tabbed plug-in developed for the NCI-specific editing environment. This plug-in provides additional editing capabilities that are not available in the Protégé OWL plug-in. To facilitate use of the NCI Protégé Plug-In, the Ontology Description Logic (DL) used by Apelon's Terminology Development Environment has been converted to OWL.

The NCI Protégé Plug-In offers many useful features, such as

- A simplified **NCI Editor** tab that facilitates consistent editing and workflow
- An **Advanced Query** tab for enhanced searching
- A policy manager to facilitate access control.

## About the NCI Editor Tab

The main Protégé window has a tabbed interface. The top row of tabs appears just beneath the menu bar. When you log in to Protégé, the **NCI Editor** tab is the first tab that you see on the top row. As an editor, you will probably use this tab more than any other.

The NCI Editor tab has a split pane view with the following features:

- The **Class Browser** on the left enables you to select a class or subclass and view details for that class in the **Edit** tab on the right. The Browser includes a Search field for entering search terms and a button for executing a search. For more information about the Browser, see [Class Browser](#) on page 43.
- A subset of tabs on the right enables you to perform various editing tasks. The **Edit** tab is the first in this series of tabs.

This section describes the interface and supported tasks of each NCI Editor tab.

[Figure 4.1](#) shows the Protégé window with the **NCI Editor** tab enabled. Note that the top row of tabs also includes two other NCI-specific plug-ins: the **Advanced Query** tab, which provides enhanced searching and query-building features, and the **Changes** tab, used mainly by workflow managers but visible to all users.

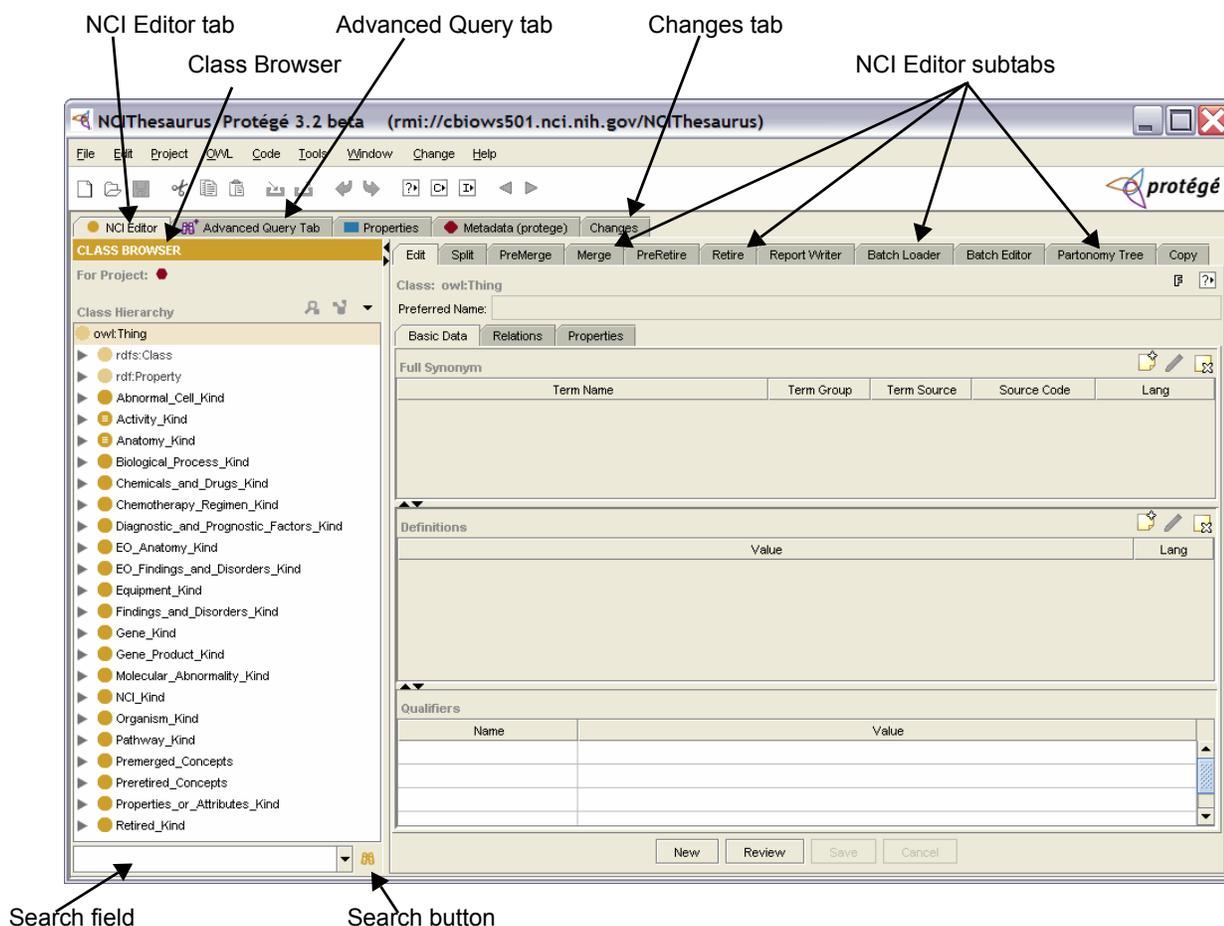


Figure 4.1 Main Protégé window with NCI Editor tab and subtabs

## Class Browser

Appearing on the left of the NCI Editor tab, the **Class Browser** (shown in [Figure 4.2](#)) provides a tree view of all available superclasses and their subclasses.

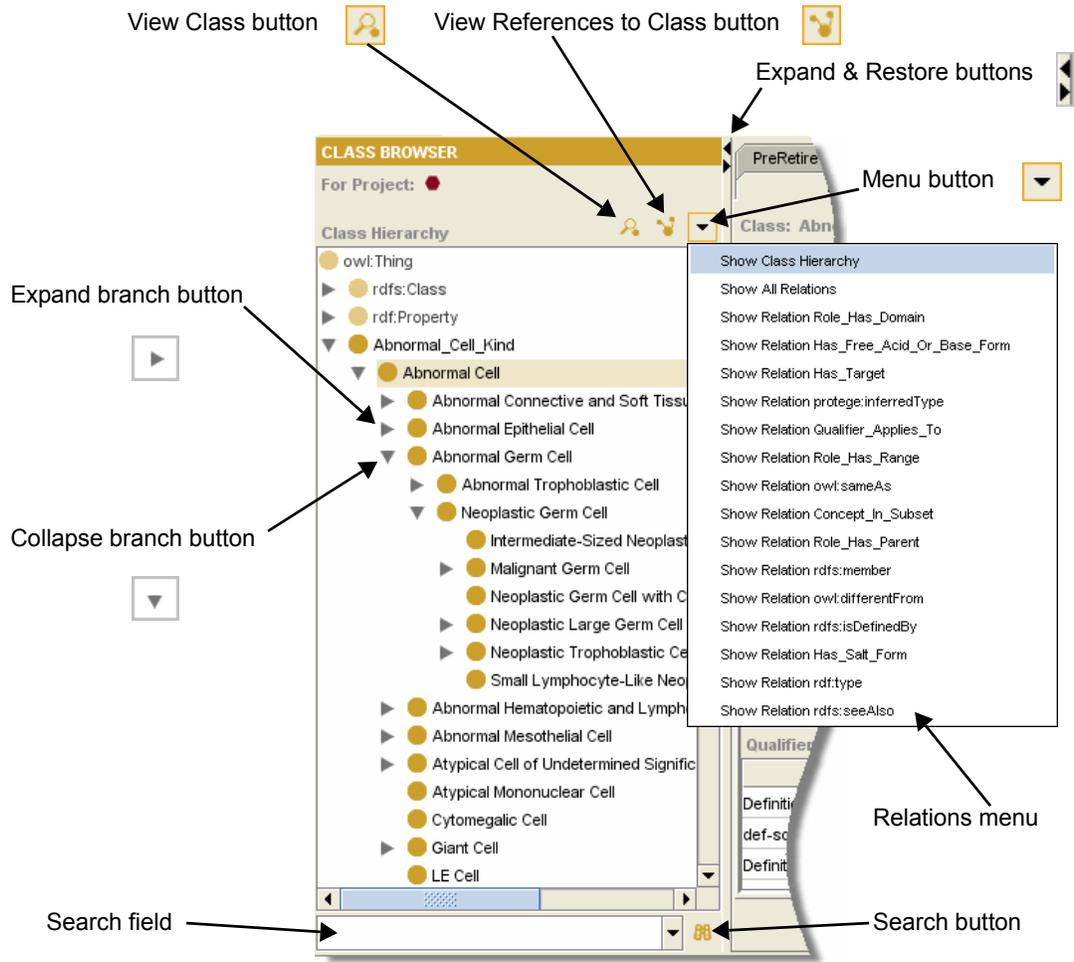


Figure 4.2 Class Browser

Table 4.1 lists the features shown in [Figure 4.2](#).

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	<b>Expand</b> button: Expands the browser to fill the entire window. <b>Restore</b> button: Restores the browser to the left pane.
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 <b>Show Class Hierarchy</b> command.

Table 4.1 Class Browser features

<b>Feature</b>	<b>Description</b>
Expand/Collapse Branch buttons	Shows/hides subclasses of a selected class.
Search field/Search button 	Enables you to type a search string and execute a search.

Table 4.1 Class Browser features

## From here...

The rest of this section discusses each of the NCI Editor tabs and related features:

- [Edit Tab](#) on page 45
- [Review Window](#) on page 49
- [Split Tab](#) on page 50
- [PreMerge Tab](#) on page 51
- [Merge Tab](#) on page 52
- [PreRetire Tab](#) on page 53
- [Retire Tab](#) on page 54
- [Report Writer Tab](#) on page 55
- [Batch Loader tab](#) on page 56
- [Batch Editor Tab](#) on page 57
- [Partonomy Tree Tab](#) on page 58
- [Copy Tab](#) on page 59

---

**Note:** If a button has an icon but no text description, its picture is included in the table. If a button has only a text label (such as **Save**), its picture is not included.

---

## Edit Tab

The **Edit** tab is used to maintain annotation and relation data for a selected class. The class data appears on three subtabs: **Basic Data**, **Relations**, and **Properties**.

[Figure 4.3](#) shows the overall layout of the **Edit** tab, and [Table 4.2](#) lists its features. The next three sections discuss the three subtabs in more detail.

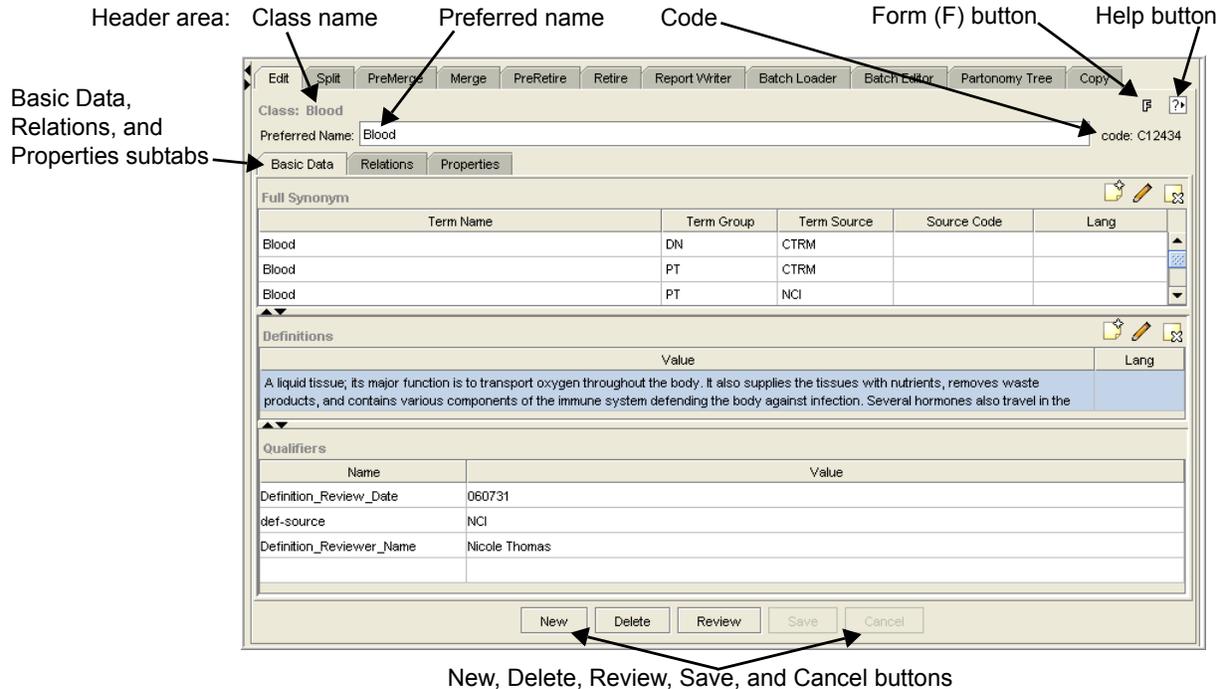


Figure 4.3 Edit tab

Feature	Description
Header area	Shows the class name, preferred name, and code.
Form button 	Used by administrators to set up Protégé projects on the server.
Help button 	Provides access to help and reference information.
Basic Data, Relations, and Properties subtabs	Show detailed information for the class. For more information, see the following sections: <ul style="list-style-type: none"> <li>• <a href="#">Basic Data Subtab</a> on page 46</li> <li>• <a href="#">Relations Subtab</a> on page 47</li> <li>• <a href="#">Properties Subtab</a> on page 48</li> </ul>
New button	Creates a new subclass.
Delete	Deletes the selected subclass.
Review button	Opens a window that shows changes made to a class, even before you have saved them. See <a href="#">Review Window</a> on page 49.
Save button	Saves your changes.
Cancel button	Discards unsaved changes.

Table 4.2 Edit tab features

## Basic Data Subtab

The **Basic Data** subtab has three panels that show, respectively, full synonym annotation properties, definition annotation properties, and definition qualifiers.

*Figure 4.4* shows the layout of this subtab and its panels, and *Table 4.3* lists its features. For more information about working with this tab, see *Creating a Class* on page 94.

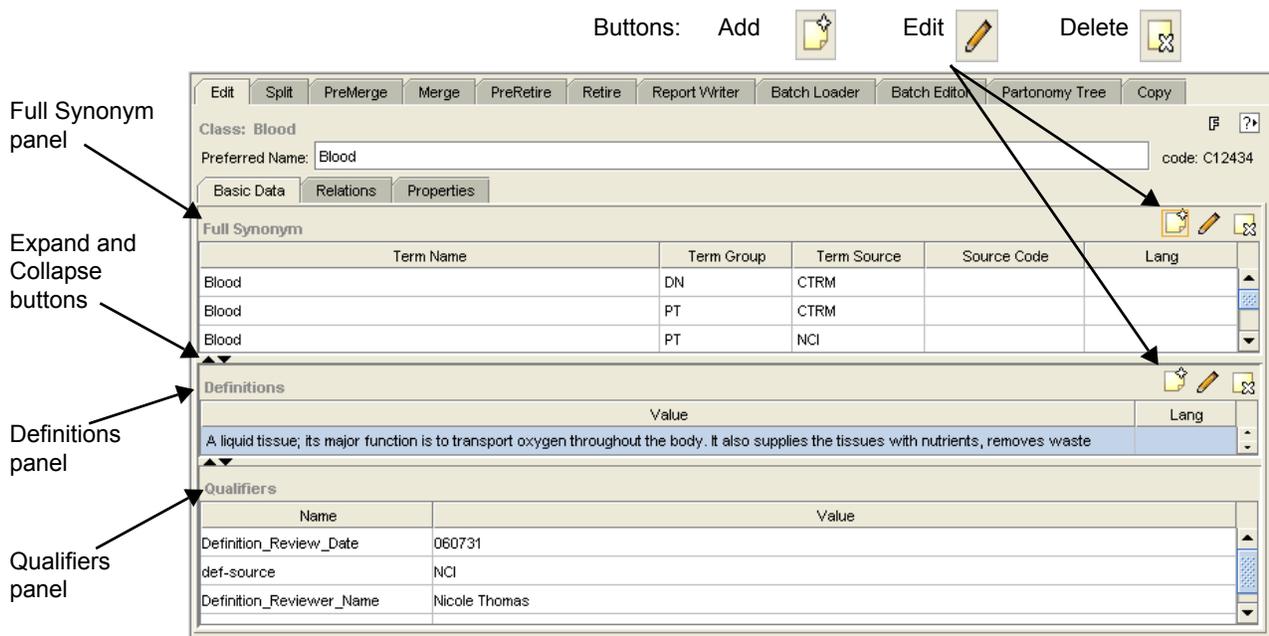


Figure 4.4 Basic Data subtab

Feature	Description
Add button 	Opens a window for creating a full synonym or definition.
Edit button 	Opens a window for editing a selected full synonym or definition, depending on what you select.
Delete button 	Deletes a selected full synonym or property. A prompt appears before the item is deleted.
Full Synonym panel	Shows full synonym values and enables you to create, edit, and delete full synonym annotation properties.
Definitions panel	Shows definition values and enables you to create, edit, and delete definition properties.
Qualifiers panel	Shows qualifier values for definition properties. To edit the <i>def-source</i> or <i>attr</i> values, click the <b>Edit</b> button in the Definitions panel. When you save the changes, the <i>Definition_Review_Date</i> and <i>Definition_Reviewer_Name</i> properties update automatically.

Table 4.3 Basic Data subtab features

## Relations Subtab

The **Relations** subtab shows values for restrictions and parent classes. It also shows properties and values for associations.

[Figure 4.5](#) shows the layout of the Relations subtab, and [Table 4.4](#) lists its features. For more information about working with this tab, see [Treeing Classes](#) on page 98, [Asserting Relations](#) on page 107, and [Adding an Association](#) on page 119.

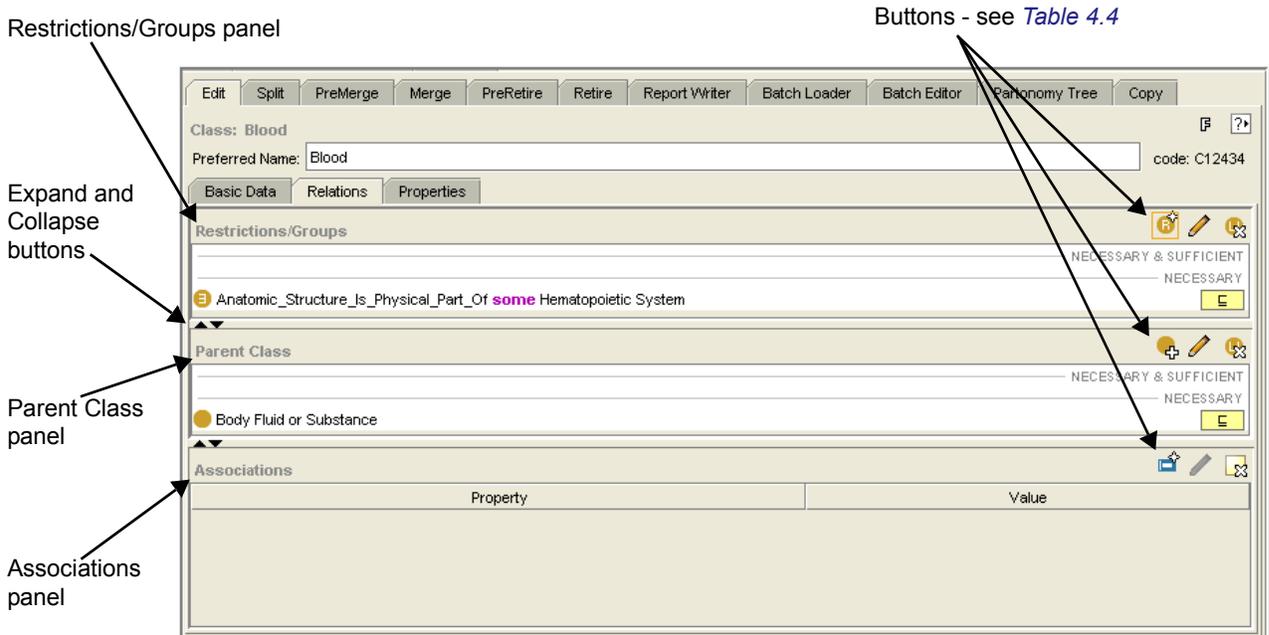


Figure 4.5 Relations subtab

Feature	Description
Restrictions/Groups panel	Shows restriction values and enables you to create, edit, and delete them.
Add, Edit, and Delete Restriction/Group buttons	<ul style="list-style-type: none"> <li>Opens a window for creating a new restriction. </li> <li>Opens a window for editing a selected restriction. </li> <li>Deletes a selected row. </li> </ul>
Parent Class panel	Shows parent classes and enables you to add, edit, and delete them.
Add, Edit (Modify), and Delete Parent Class buttons	<ul style="list-style-type: none"> <li>Opens a window for adding a new parent class. </li> <li>Opens a window for editing a selected parent class. </li> <li>Deletes a selected parent class. </li> </ul>
Associations panel	Displays object-valued properties and enables you to add, edit, or delete them.
Add, Edit, and Delete Object-Valued Property buttons	<ul style="list-style-type: none"> <li>Opens a window for adding an object-valued property. </li> <li>Opens a window for editing an object-valued property. </li> <li>Deletes a selected object-valued property. </li> </ul>

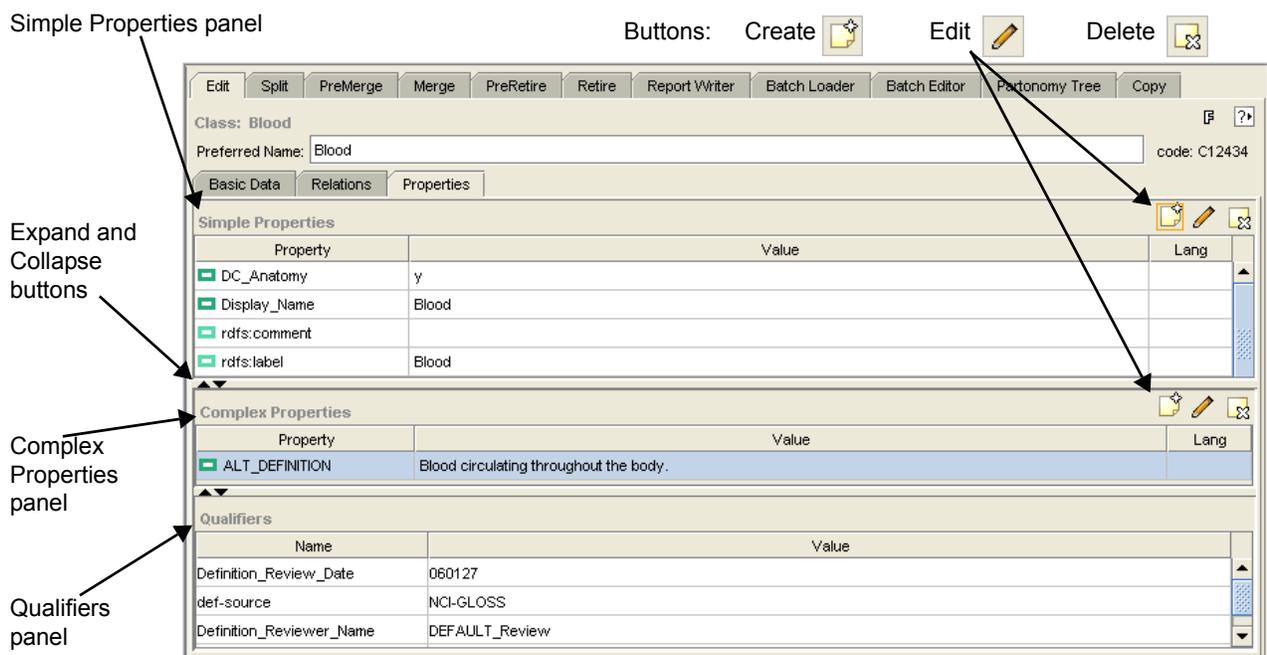
Table 4.4 Relations subtab features

## Properties Subtab

The **Properties** subtab shows simple properties, complex properties, and qualifier values for complex properties.

**Note:** You can find properties on both this subtab and on the Basic Data subtab (page 46).

*Figure 4.6* shows the layout of this subtab, and *Table 4.5* lists its features.



*Figure 4.6 Properties subtab*

Feature	Description
Simple Properties panel	Shows simple property values and enables you to create, edit, and delete them.
Create, Edit, and Delete Simple Properties buttons	<ul style="list-style-type: none"> <li>Opens a window for selecting a new property. </li> <li>Opens a window for editing an existing property. </li> <li>Deletes a selected property.  A prompt appears before the property is deleted.</li> </ul>
Complex Properties panel	Shows complex property values and enables you to create, edit, and delete them.
Create, Edit, and Delete Property buttons	<ul style="list-style-type: none"> <li>Opens a window for selecting a new property. </li> <li>Opens a window for editing existing properties. </li> <li>Deletes a selected item.  A prompt appears before the property is deleted.</li> </ul>
Qualifiers panel	Shows qualifier values for complex properties. To edit a value, click the <b>Edit property</b> button in the Complex Properties panel.

*Table 4.5 Properties subtab features*

## Review Window

When you use the three subtabs of the **Edit** tab (Basic Data, Relations, and Properties) to edit data, you can freely switch between those subtabs without having to click the **Save** button each time. Protégé holds your changes in memory as long as you continue working with data shown on the Edit tab.

**Note:** If you switch to another parent tab (such as Split, PreMerge, or Report Writer) without saving your changes, a Confirmation window prompts you to save. Your changes are discarded unless you answer **Yes** to the prompt.

Before saving changed data, you can view a data summary by clicking the **Review** button. This opens a convenient Review window that summarizes data for the current class, including changes that you have made but have not yet saved.

Use this window often to review unsaved changes.

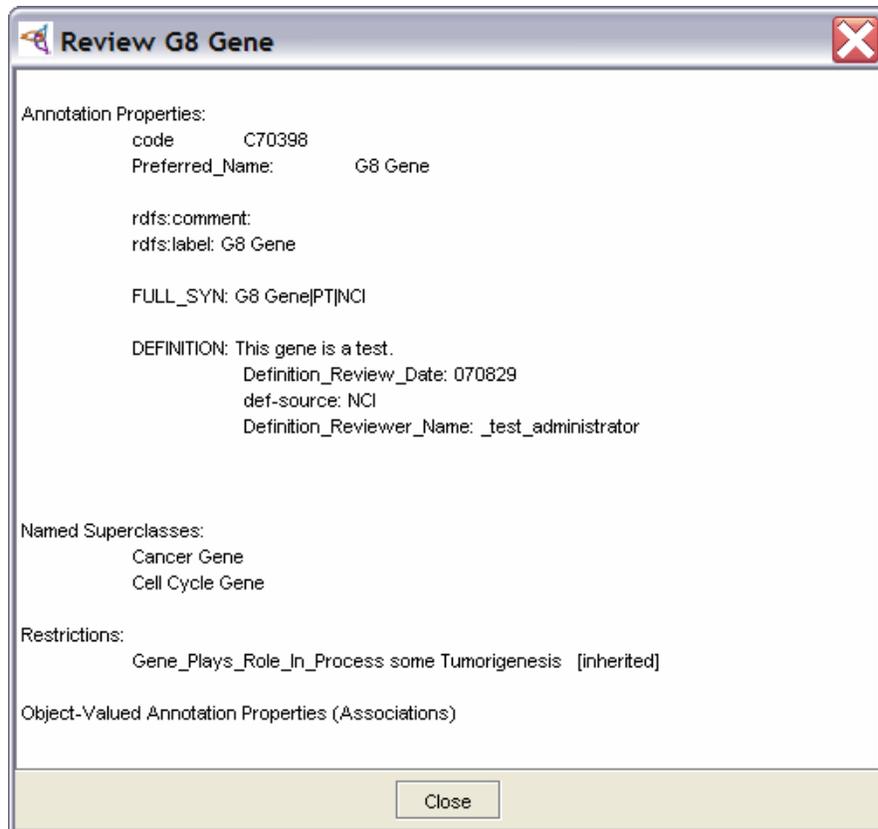


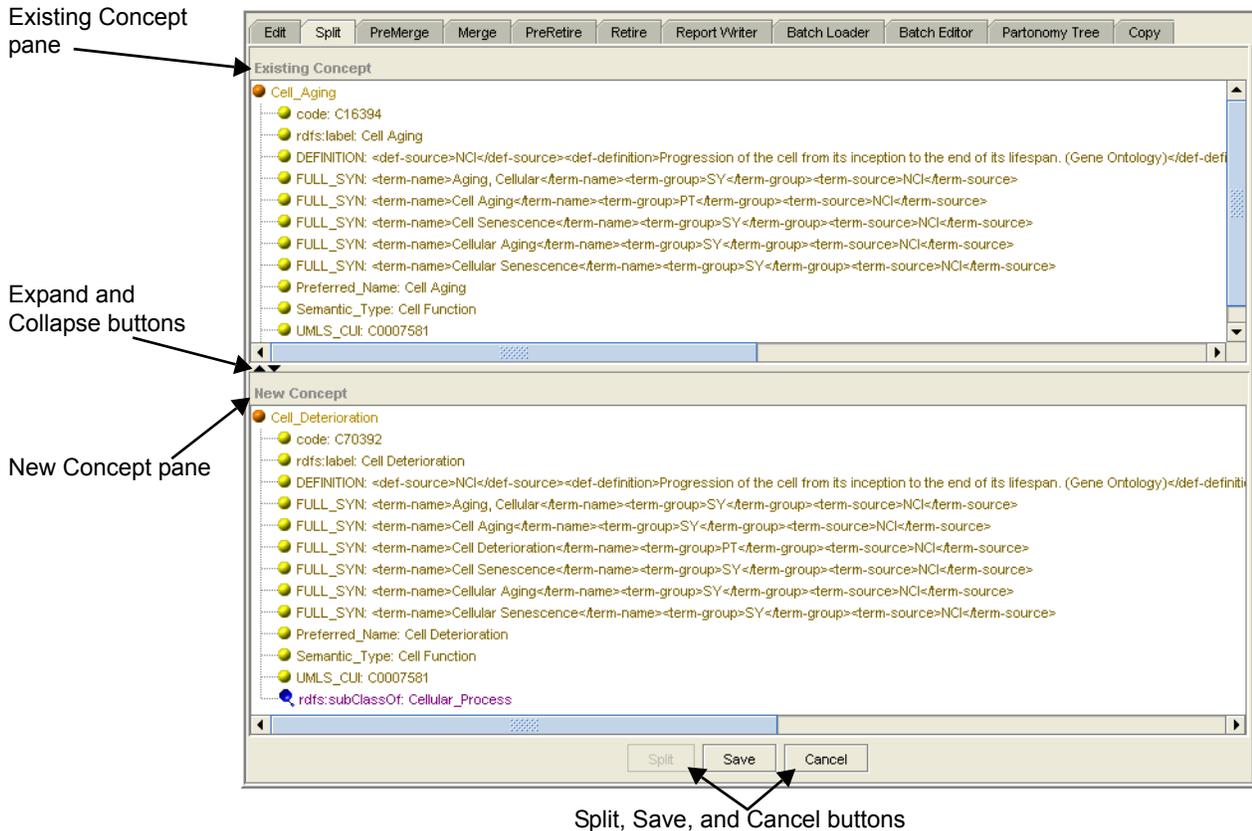
Figure 4.7 Review window

## Split Tab

**Note:** To view this tab, select a class in the Class Browser.

The **Split** tab enables you to create a new class from an existing class. This tab includes upper and lower horizontal panes (respectively, existing and new concepts). To split a class, drag it from the Class Browser into the upper pane, then click the **Split** button. For complete instructions, see *Splitting a Class* on page 124.

*Figure 4.8* shows the layout of this tab, and *Table 4.6* lists its features.



*Figure 4.8* Split tab

<b>Feature</b>	<b>Description</b>
Existing Concept pane (upper)	Shows a tree representation of a selected class.
New Concept pane (lower)	Shows a tree representation of a new class that has been created from an existing class.
Split button	Creates a new class based on the class shown in the Existing Concept pane.
Save button	Saves the newly created class.
Cancel button	Discards unsaved data.

*Table 4.6* Split tab features

## PreMerge Tab

**Note:** To view this tab, select a class in the Class Browser.

The **PreMerge** tab enables you to flag two classes for a merge. Although you can flag classes, only workflow administrators can perform the actual merge. This tab includes upper and lower horizontal panes (respectively, surviving and retiring concepts).

To flag two classes, drag the surviving class into the upper pane, drag the retiring class into the lower pane, then click the **PreMerge** button. For complete instructions, see [Merging Classes](#) on page 127.

[Figure 4.9](#) shows the layout of this tab, and [Table 4.7](#) lists its features.

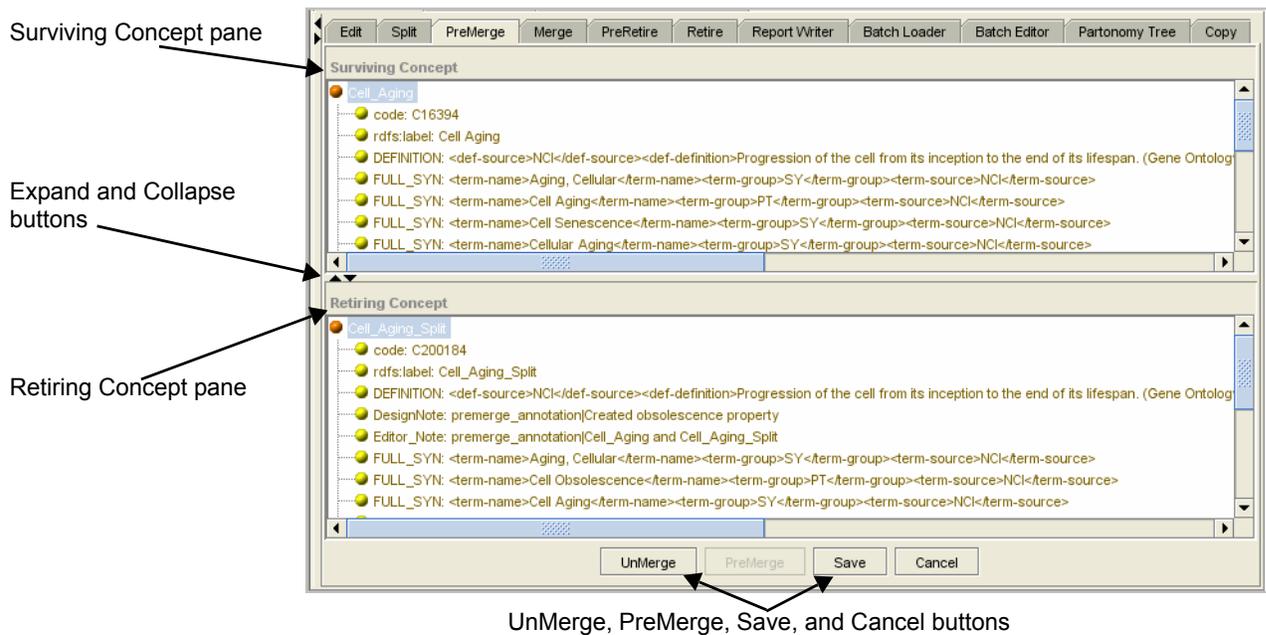


Figure 4.9 PreMerge tab

Feature	Description
Surviving Concept pane	Shows a tree representation of the <i>surviving</i> concept.
Retiring Concept pane	Shows a tree representation of the <i>retiring</i> concept.
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: <ul style="list-style-type: none"> <li>Surviving = <i>Merge_Source</i></li> <li>Retiring = <i>Merge_Target</i></li> </ul> <b>Note:</b> The value of each property is the code of the referenced class.
Cancel button	Reverses a pre-merge operation.

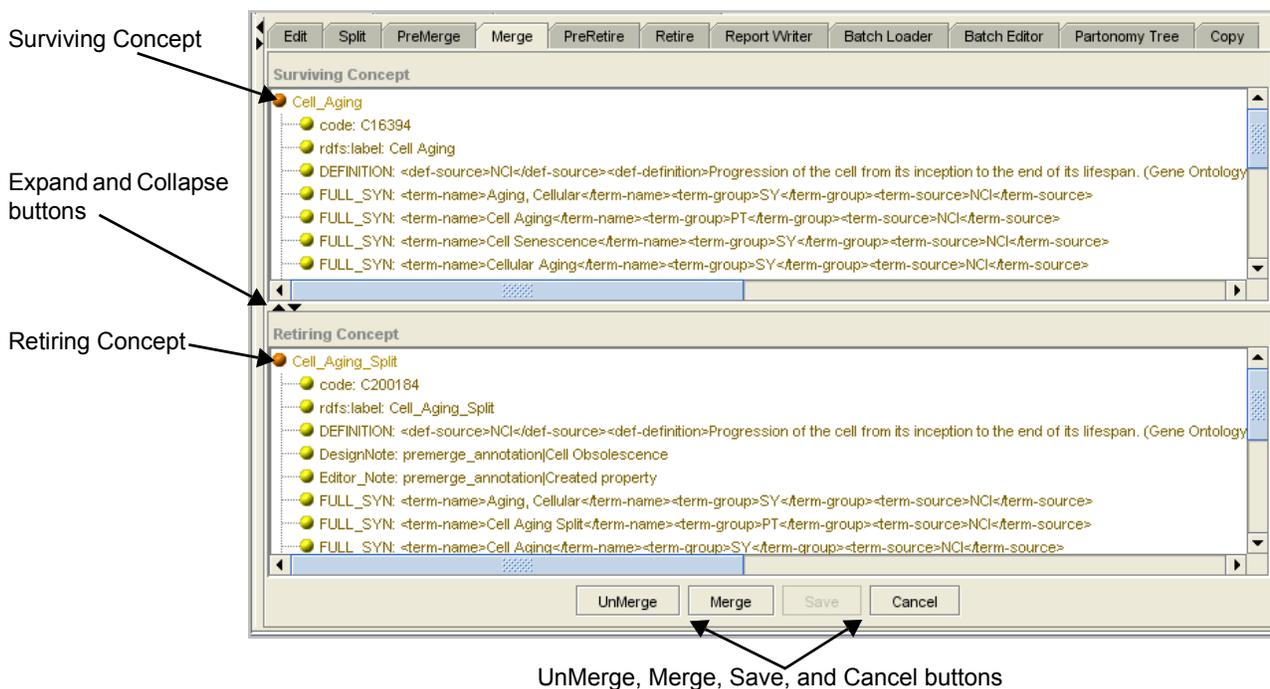
Table 4.7 PreMerge tab features

## Merge Tab

**Note:** To view this tab, select a class in the Class Browser.

The **Merge** tab enables workflow managers to merge two classes that have been flagged using the pre-merge action (see *PreMerge Tab* on page 51). For complete instructions, see *Merging Classes* on page 127.

*Figure 4.10* shows the layout of the Merge tab, and *Table 4.8* lists its features.



*Figure 4.10 Merge tab*

<b>Feature</b>	<b>Description</b>
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 4.8 Merge tab features*

## PreRetire Tab

**Note:** To view this tab, select a class in the Class Browser.

The **PreRetire** tab enables you to flag a class for retirement. This tab includes two subtabs that show, respectively, subclasses and referencing classes. For complete instructions, see [Pre-Retire: Flagging a Class for Retirement](#) on page 131.

[Figure 4.11](#) shows the layout of this tab, and [Table 4.9](#) lists its features.

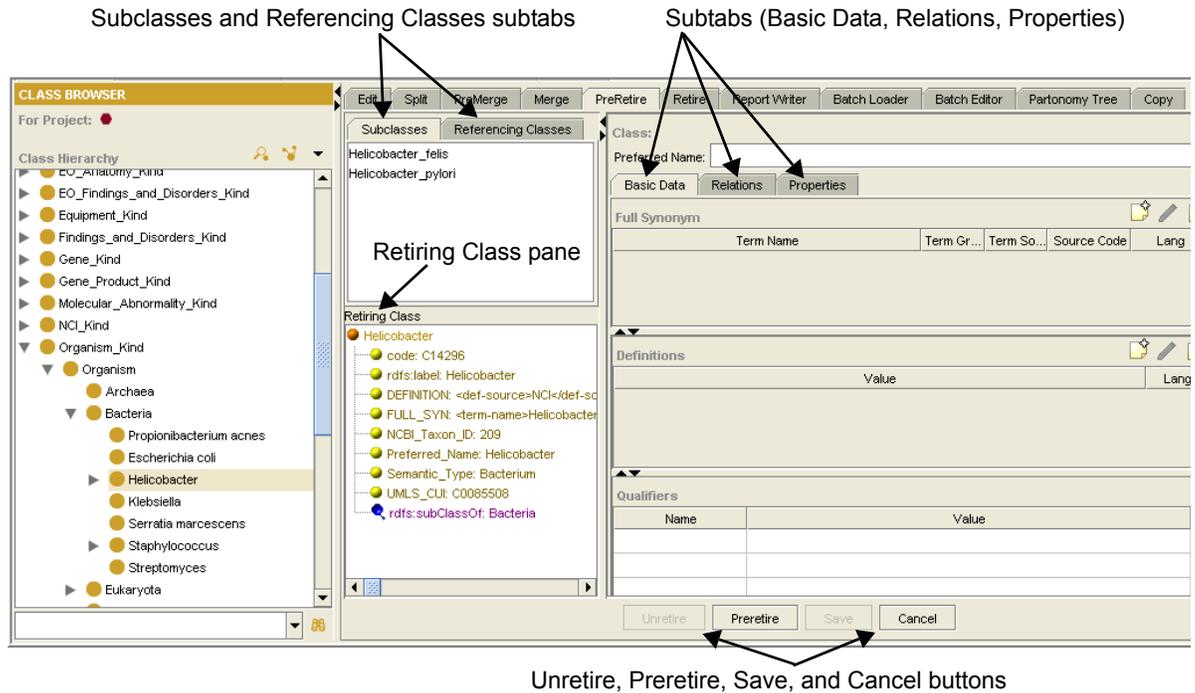


Figure 4.11 PreRetire tab

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.
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 4.9 PreRetire tab features

## Retire Tab

**Note:** To view this tab, select a class in the Class Browser.

The **Retire** tab enables authorized users (e.g., workflow managers) to retire a class that has been flagged by a pre-retire action (see *PreRetire Tab* on page 53). This tab has a single pane that shows a tree representation of the selected class. You can drag a pre-retired class from the *Preretired Concepts* branch in the Class Browser and drop it into this pane. For complete instructions, see *Retiring a Flagged Class (Workflow Managers Only)* on page 135.

Figure 4.12 shows the layout of this tab, and Table 4.10 lists its features.

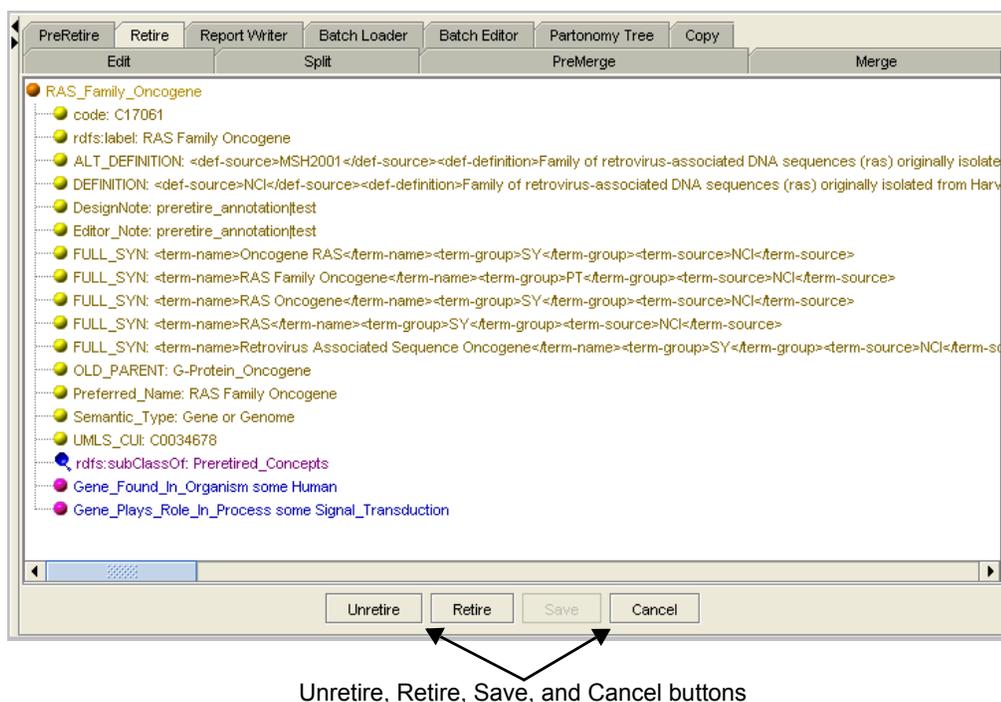


Figure 4.12 Retire tab

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 <i>pre-retired</i> to <i>retired</i> .
Cancel button	Discards unsaved changes.

Table 4.10 Retire tab features

## Report Writer Tab

**Note:** To view this tab, select a class in the Class Browser.

The **Report Writer** tab enables you to produce reports for selected classes. You can choose to generate a report with or without class attributes.

As shown in [Figure 4.13](#), this tab has a very simple interface. It is used only to open the Report Writer window, shown in the same figure. In the window, you can specify the name and location of an external report file (usually a text file). After generating the report file from the window, you can view the file 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 tasks 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 task.

For complete instructions, see [Using the Report Writer](#) on page 136.

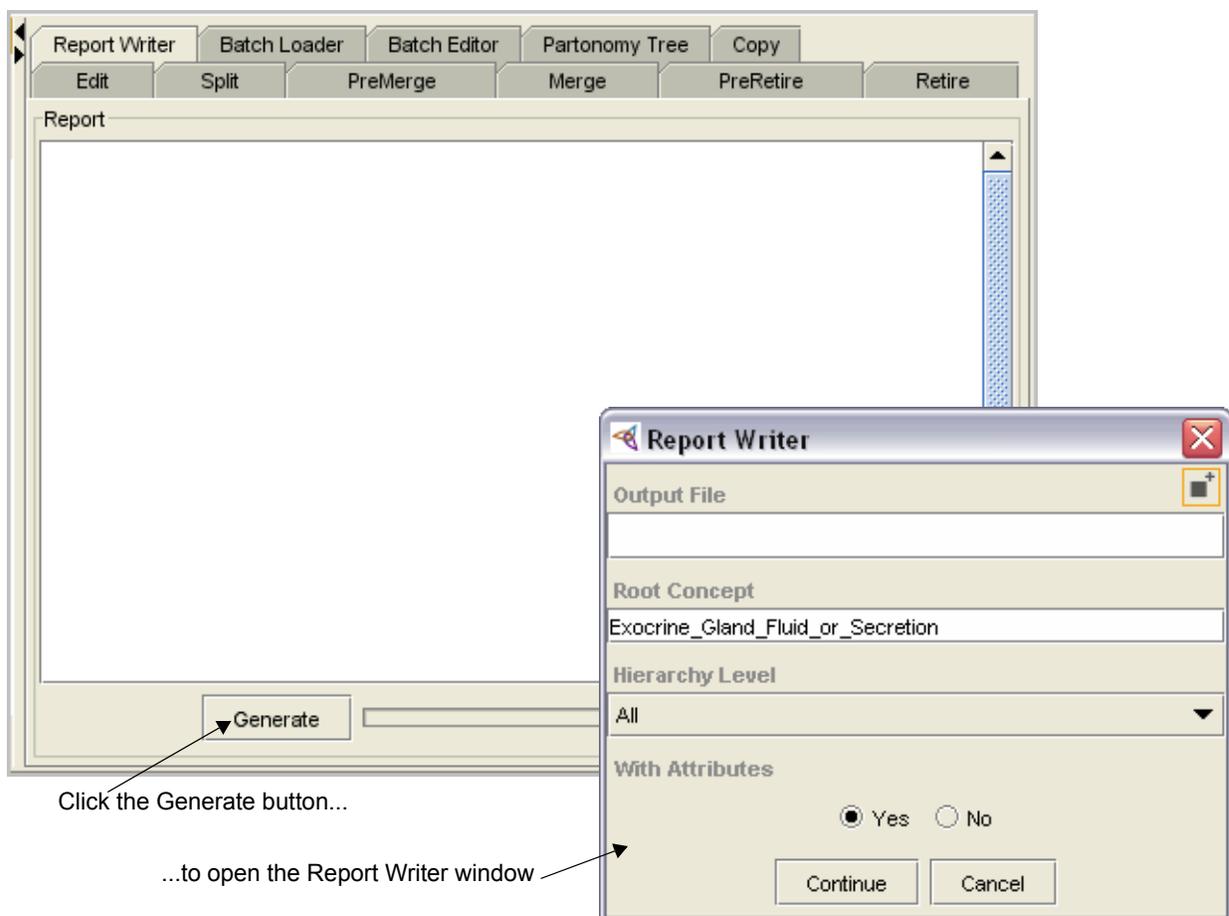


Figure 4.13 Report Writer tab and Report Writer window

## Batch Loader tab

**Note:** To view this tab, select a class in the Class Browser.

The **Batch Loader** tab enables you to load a batch of classes for editing. The Log area of the tab is empty until you specify and load an input file.

Protégé accepts a tab-delimited (text) input file using a specified format. For more information about the format specifications, see [Loading a Batch of Classes for Editing](#) on page 142.

[Figure 4.14](#) shows the Batch Loader tab with a sample output file displayed, and [Table 4.11](#) lists its features.

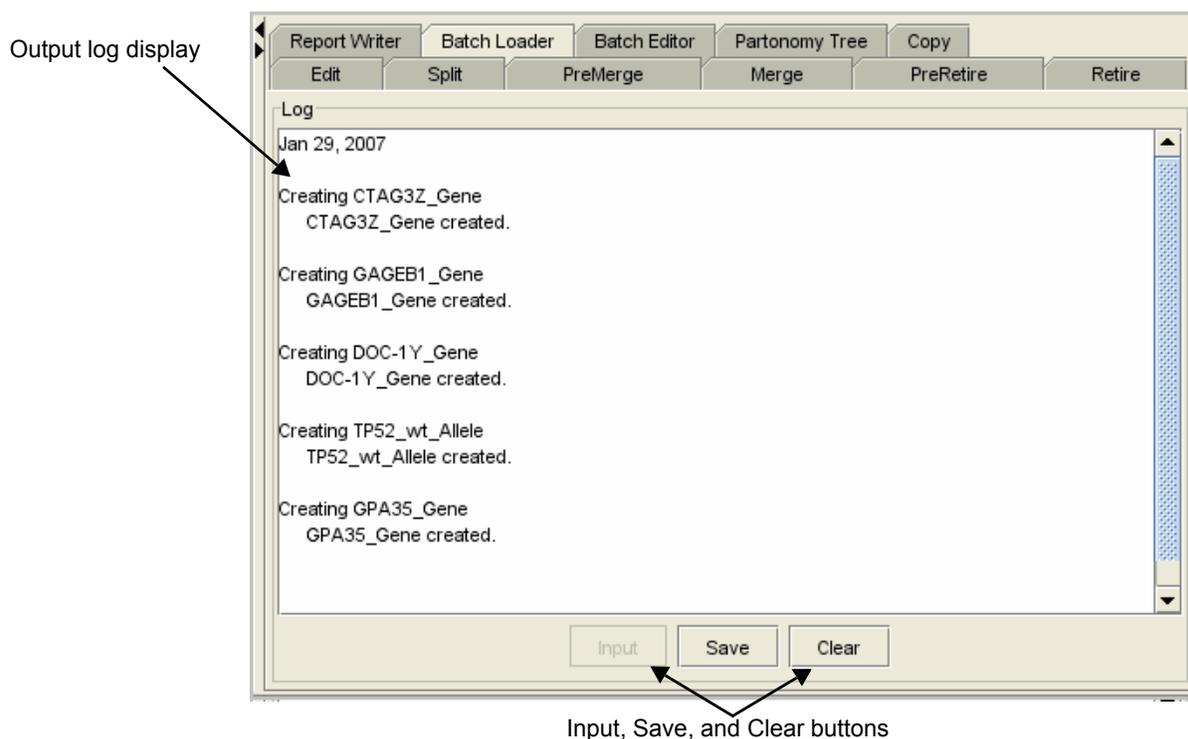


Figure 4.14 Batch Loader tab

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 <b>Clear</b> 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.

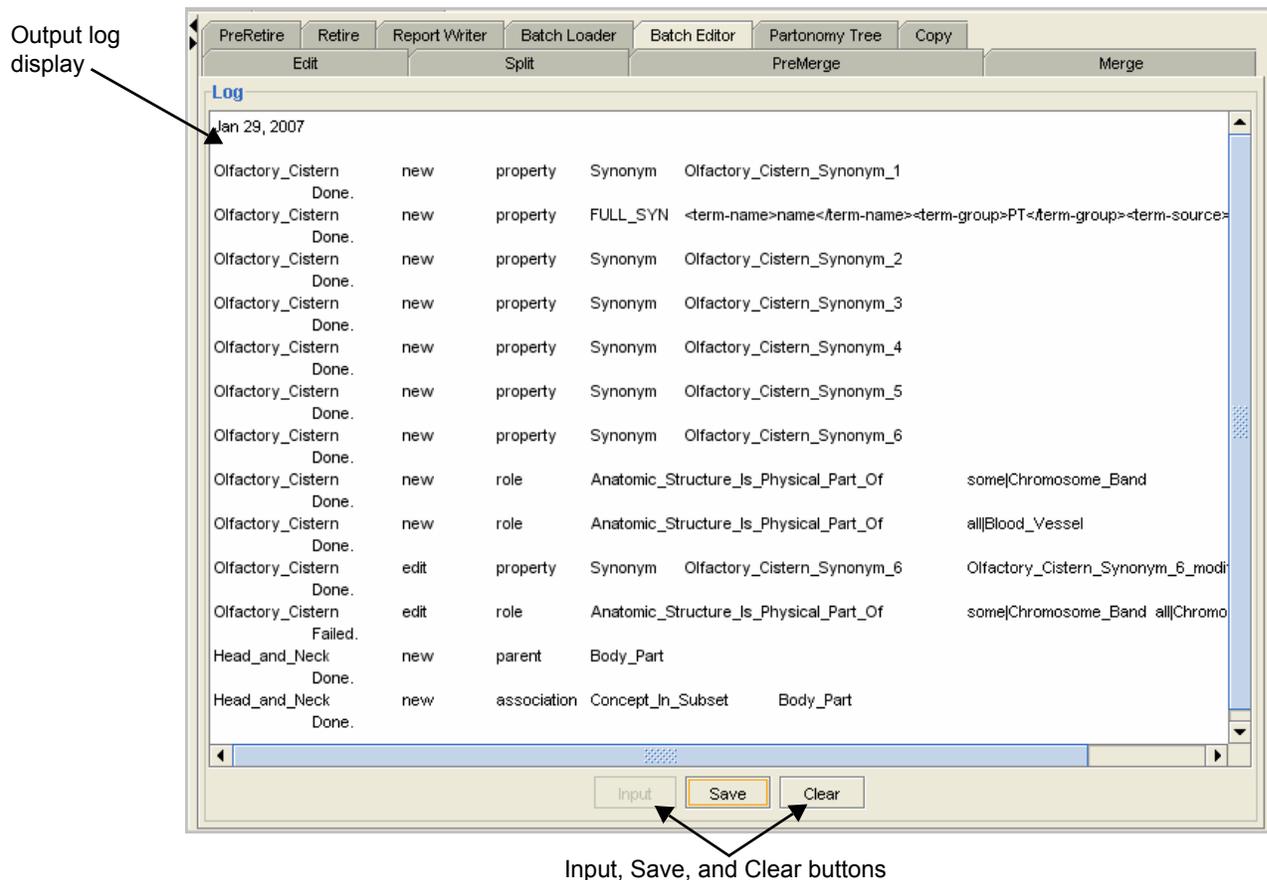
Table 4.11 Batch Loader tab features

## Batch Editor Tab

**Note:** To view this tab, select a class in the Class Browser.

The **Batch Editor** tab enables you to submit a batch of edits to the knowledge base by loading a tab-delimited input file in a specific format. For more information on the file format specifications, see *Editing a Batch of Classes* on page 146.

*Figure 4.15* shows the layout of this tab, and *Table 4.12* lists its features.



*Figure 4.15* Batch Editor tab

<b>Feature</b>	<b>Description</b>
Output log display	Shows the log status (the status of the batch edit 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.

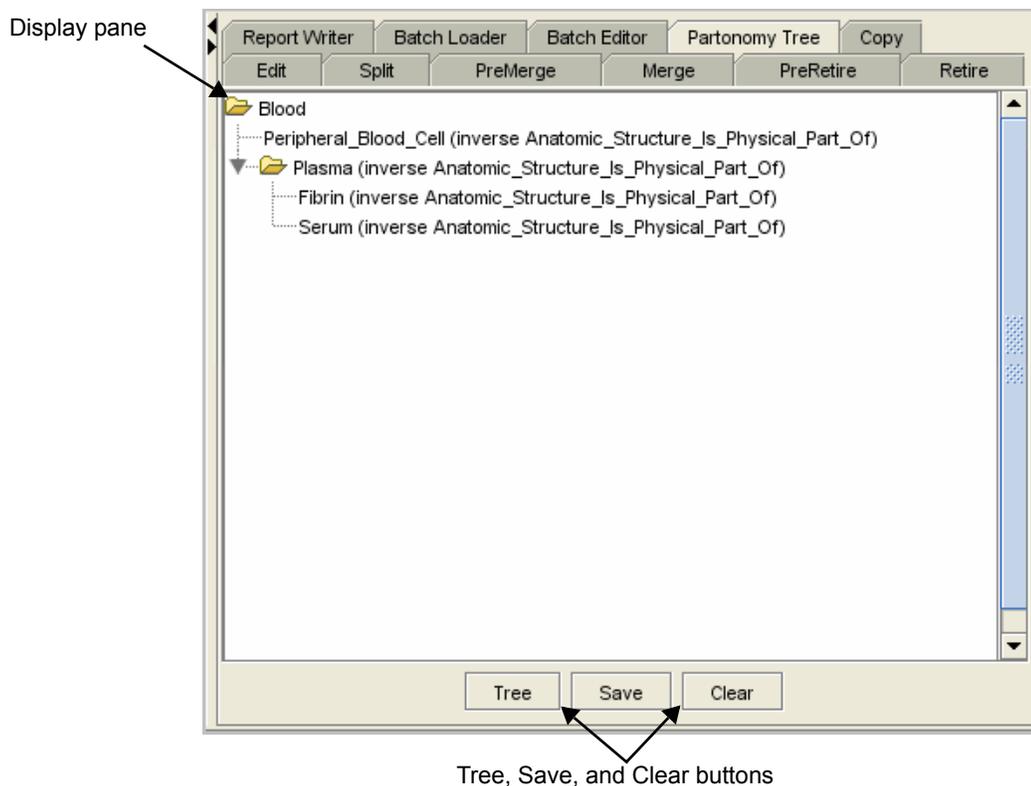
*Table 4.12* Batch Editor tab features

## Partonomy Tree Tab

**Note:** To view this tab, select a class in the Class Browser.

The **Partonomy Tree** tab enables you to select a root class and display it as a partonomy tree. A partonomy tree shows classes that are connected by *part\_of* relations.

*Figure 4.16* shows the layout of the Partonomy Tree tab, and *Table 4.13* lists its features. For complete instructions on using this tab, see *Generating a Partonomy Tree* on page 150.



*Figure 4.16 Partonomy Tree tab*

<b>Feature</b>	<b>Description</b>
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 4.13 Partonomy Tree tab feature*

## Copy Tab

**Note:** To view this tab, select a class in the Class Browser.

The **Copy** tab enables you to clone a class from an existing class. Simply drag a class from the Class Browser into the upper pane, then click the **Clone** button to copy the class. The new class appears in the lower pane with a newly assigned code. For more information and complete instructions, see [Copying a Class](#) on page 152.

The two-pane view on the Copy tab facilitates simultaneous editing of two classes. Select an object, then right-click to select from a number of editing commands.

[Figure 4.17](#) shows the layout of this tab, and [Table 4.14](#) lists its features.

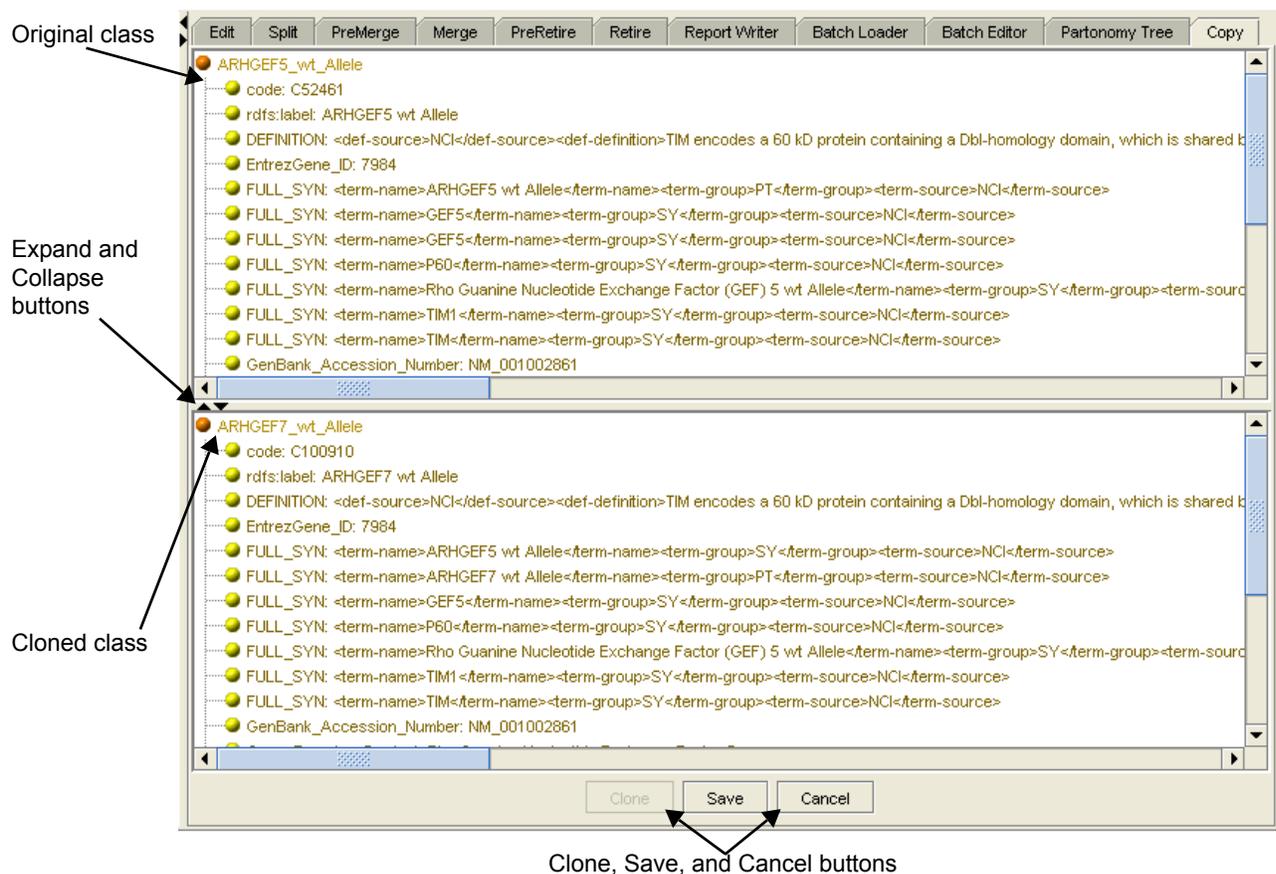


Figure 4.17 Copy tab

Feature	Description
Upper and lower panes	Shows original and cloned classes.
Clone button	Creates a clone of the selected class.
Save button	Saves the newly cloned.
Cancel button	Cancels the copy procedure and clears both panes.

Table 4.14 Copy tab features

## About the Advanced Query Tab

The **Advanced Query** tab offers fine-tuned control over the scope and complexity of searches:

- You can search on a class name or on any property, including FULL\_SYN, DEFINITION, or preferred name.
- You can build search expressions from several search parameters such as *contains*, *starts with*, or *ends with*.
- You can narrow or broaden the search scope by selecting a **Match All** (AND) or **Match Any** (OR) option.

This section introduces you to the Advanced Query tab interface. For more detailed information about searching and building queries, see [Searching: Simple and Advanced](#) on page 67.

[Figure 4.18](#) shows the layout of the Advanced Query tab, and [Table 4.15](#) (page 61) lists its features.

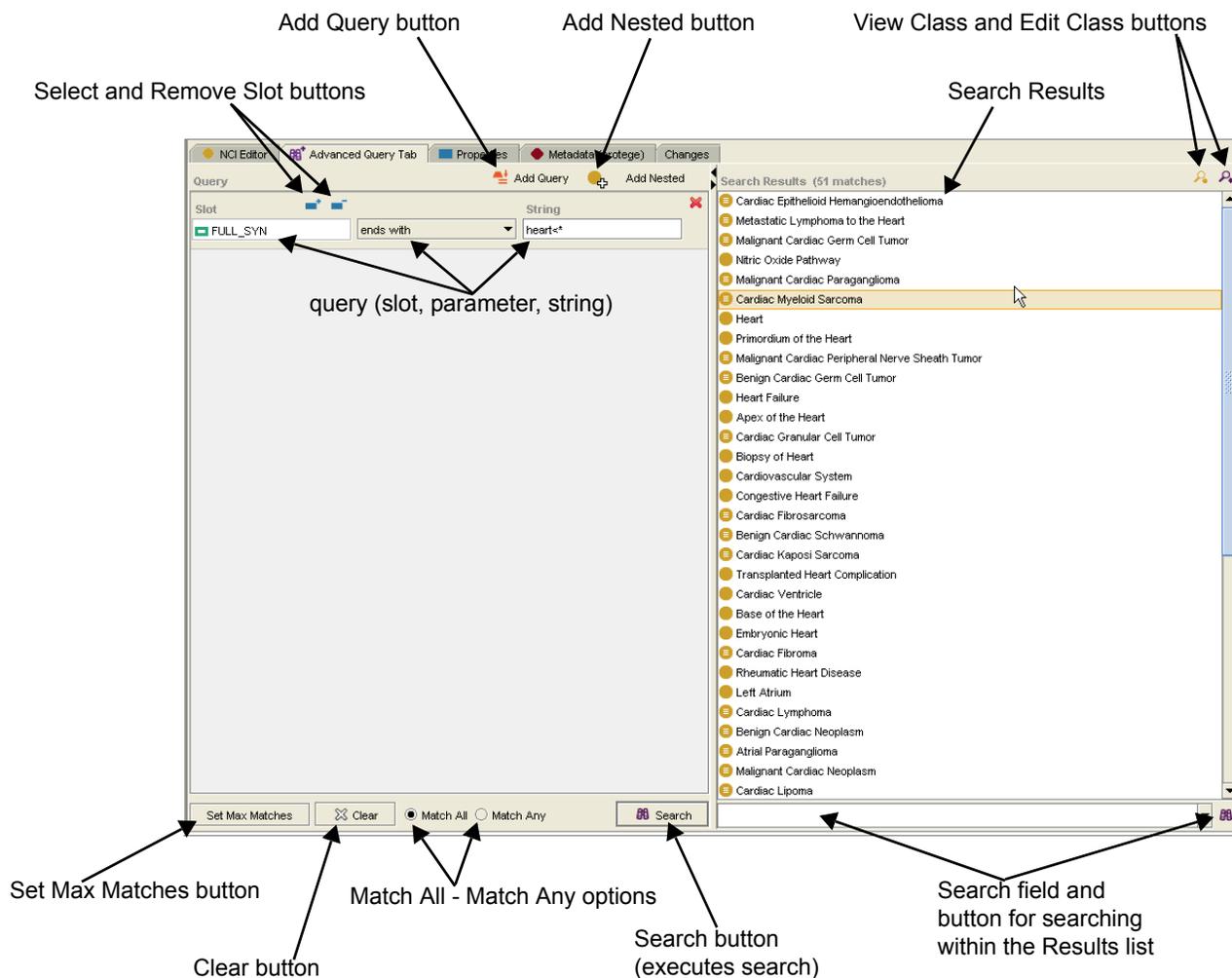


Figure 4.18 Advanced Query tab

Table 4.15 lists the Advanced Query tab features.

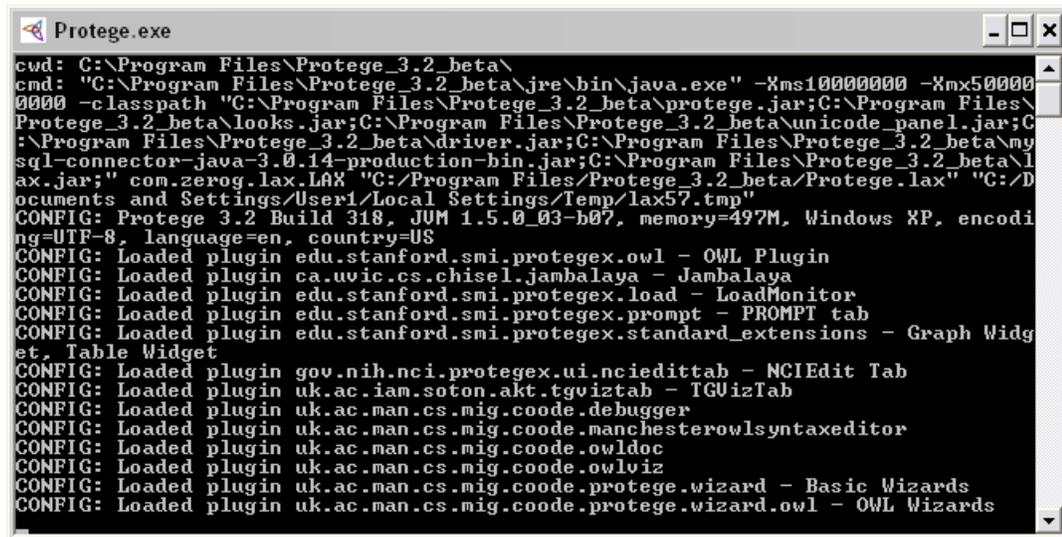
Feature	Description
Add Query button 	Adds a new query, including a Slot field (with Preferred_Name displayed), a Search Parameters list, and an empty String field. Use these tools to build your own expressions.
Add Nested button 	Enables you to build a query for finding classes with specific restrictions or associations.
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 Search Parameters list, and removes the String field. To restore these features and add new query information, click the <b>Select Slot</b> button.
Query	Shows a full search expression: slot, parameters ( <i>contains</i> , <i>starts with</i> , <i>ends with</i> , <i>exact match</i> , or <i>sounds like</i> ), and a typed string. 
Set Max Matches 	Opens a window in which you can set the maximum number of matches to be retrieved.
Clear button 	Removes all expressions and leaves only one generic expression: <i>Preferred_Name contains &lt;blank&gt;</i> . This button does not affect the Search Results pane on the right.
Match All / Match Any options 	Enable you to control the scope of a search. <b>Match All</b> (the default) performs an AND search, and <b>Match Any</b> performs an OR search.
Search button 	Executes a search.
Edit Class button 	Shows the basic data, relations, and properties of a selected search result in the Edit tab.
View Class button 	Opens a read-only window in which you can view basic data, relations, and properties for a selected search result.
Search Results list	Lists the returned results from a search and enables you to search within the list. See <a href="#">Figure 4.18</a> .
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 search button on the right to execute the search. 

Table 4.15 Advanced Query tab features

## About the Console

When you launch 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 remains open in the background each time you run a Protégé session.

*Figure 4.19* shows the Console as it appears after Protégé first opens.



```

Protege.exe
cmd: C:\Program Files\Protege_3.2_beta\
cmd: "C:\Program Files\Protege_3.2_beta\jre\bin\java.exe" -Xms100000000 -Xmx500000000 -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\mysql-connector-java-3.0.14-production-bin.jar;C:\Program Files\Protege_3.2_beta\lax.jar;" com.zerog.lax.LAX "C:/Program Files/Protege_3.2_beta/Protege.lax" "C:/Documents and Settings/User1/Local Settings/Temp/lax57.tmp"
CONFIG: Protege 3.2 Build 318, JVM 1.5.0_03-b07, memory=497M, Windows XP, encoding=UTF-8, language=en, country=US
CONFIG: Loaded plugin edu.stanford.smi.protege.owl - OWL Plugin
CONFIG: Loaded plugin ca.uvic.cs.chisel.jambalaya - Jambalaya
CONFIG: Loaded plugin edu.stanford.smi.protege.load - LoadMonitor
CONFIG: Loaded plugin edu.stanford.smi.protege.prompt - PROMPT tab
CONFIG: Loaded plugin edu.stanford.smi.protege.standard_extensions - Graph Widget, Table Widget
CONFIG: Loaded plugin gov.nih.nci.protege.ui.nciedittab - 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.owlloc
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 4.19 Console window*

The Console displays a line-by-line record of the actions that you perform while working with 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.

This section covers the following topics:

- [Setting the Console Display Capacity](#) on page 63

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

- [Capturing Information for Troubleshooting](#) on page 64
- [Submitting a Bug Report and File Attachment to GForge](#) on page 65

## Setting the Console Display Capacity

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, as shown in [Figure 4.20](#).

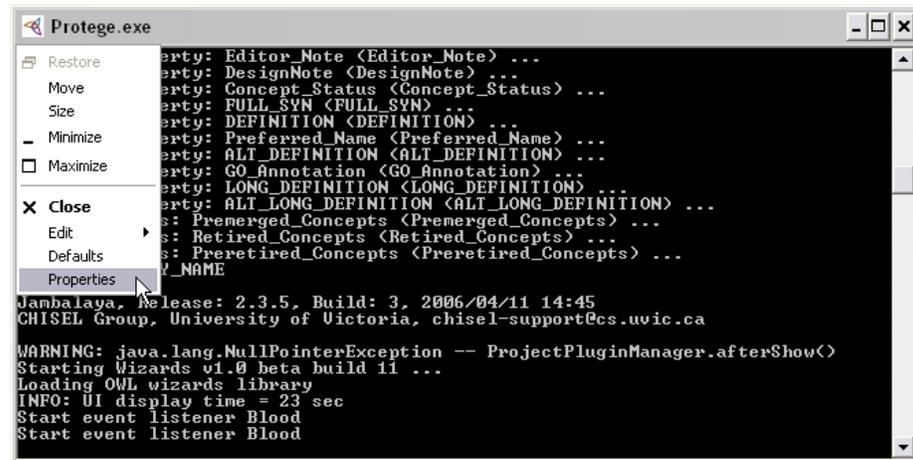


Figure 4.20 Console window menu - Properties command

3. In the Properties window, click the **Layout** tab.
4. In the **Screen Buffer Size** section, type 9999 in the **Height** field, as shown in [Figure 4.21](#). This sets the value to a maximum so that you will not lose any information that might be valuable for troubleshooting.

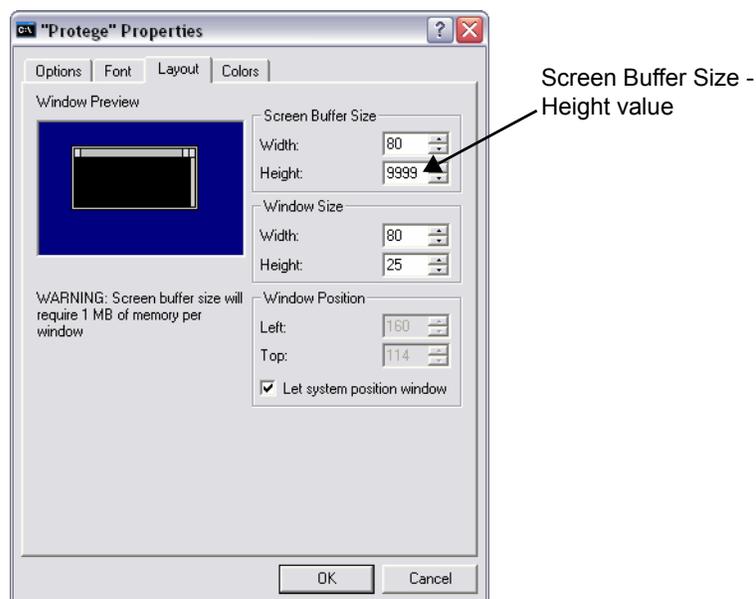


Figure 4.21 Console Properties window

5. Click **OK** to close the Console Properties window.

## Capturing Information for Troubleshooting

**Note:** Before performing this procedure, make sure that you increase the screen buffer size as directed in *Setting the Console Display Capacity* on page 63.

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

### Capturing a Thread Dump

To capture a thread dump, press CTRL + BREAK (or PAUSE BREAK, depending on your keyboard). The thread dump is appended to the content in the Console window.

### Copying the Console Window Contents

Copying the text that represents the current application state is called a *Console copy*. To perform a Console copy, 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 **Edit > Select All** to select all of the text in the window, as shown in *Figure 4.22*.

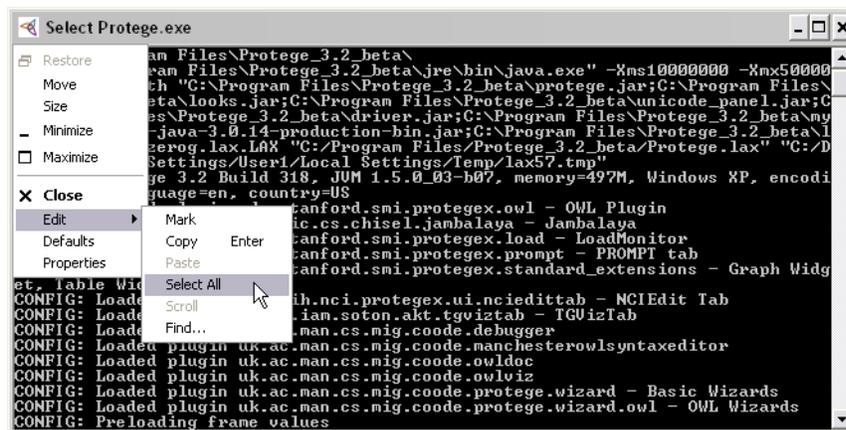


Figure 4.22 Command line window menu - Select All command

3. Using the same menu in the upper left corner of the window, select **Edit > Copy**.
4. Open a text editor such as Windows Notepad.
5. Paste the copied text into a blank text file using the menu (**Edit > Paste**) or the keyboard (CTRL + V).
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\_02-12-07.txt*).

## Submitting a Bug Report and File Attachment to GForge

Follow these steps to submit a bug tracker to the NCI Protégé Plug-In project in GForge:

1. In your Web browser's address bar, enter the following URL:  
<https://gforge.nci.nih.gov/>.
2. If you see a login prompt, enter your username and password.
3. Click the **My Page** tab.
4. On the right side, under My Projects, click **EVS Collaborative Terminology Dev Tools**.
5. Click the **Tracker** tab.
6. Click the **Protege Bugs** link.
7. On the GForge Tracker tab, click the **Submit New** link.

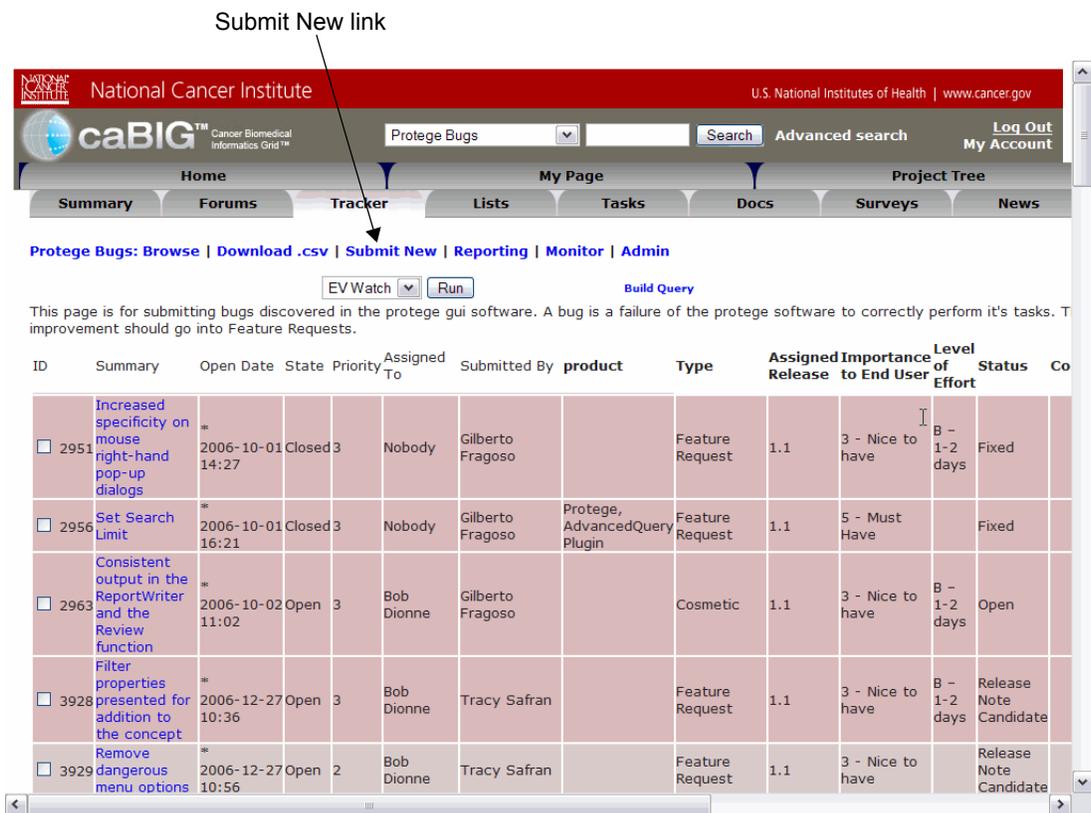


Figure 4.23 Tracker page - Submit New link

8. Complete the fields listed in [Table 4.16](#). Fields are listed across, and then down.

Field	Description
Product	Select a product name.
Type	Select a category describing the nature of the problem.
Assigned Release	Select a release number.

Table 4.16 Bug Tracker fields

<b>Field</b>	<b>Description</b>
Importance to End User	Select an importance level for the problem.
Level of Effort	Select an option describing the estimated time to resolve the problem.
Status	Assign a status indicator by selecting it from the list.
Component	Select the application component to which the problem applies.
Previous Build Tested	Enter the version number of the most recent application version used to test the problem. This number changes every time a tester verifies a developer's fix. If the status of a problem is <i>Fixed</i> , this field tells you the exact version that contains the fix.
Version Found	Enter the version of the application that always reproduces the problem. Make sure that you enter the correct version, as this field is used to track the defect.
Assigned To	Select a person's name.
Priority	Select a priority level.
Summary	Enter a summary title for the problem.
Detailed Description (with Notepad icon)	Describe the problem as thoroughly as possible. Click the Notepad icon to open a separate text entry window. Answer such questions as the following: What task were you performing when the problem happened? Were you able to capture the text of any messages that appeared?
Check to Upload & Attach File	<ol style="list-style-type: none"> <li>1. Check the box.</li> <li>2. Click the <b>Browse</b> button and find the file that you created in <a href="#">Copying the Console Window Contents</a> on page 64.</li> <li>3. Upload the file.</li> </ol>
File Description	Type a brief description of the file that you included.

Table 4.16 Bug Tracker fields (Continued)

9. Click the **Submit** button.

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

# CHAPTER 5

## SEARCHING: SIMPLE AND ADVANCED

Protégé's fine-tuned search features enable you to quickly find classes without having to browse the entire taxonomy tree. This chapter introduces you to both the simple and advanced search features that are part of the NCI Protégé Plug-In.

This chapter includes the following topics:

- [Simple vs. Advanced Searching](#) on this page
- [Performing Simple Searches](#) on page 68
- [Building Advanced Queries](#) on page 71
- [Configuring the Advanced Query Tab](#) on page 84

### Simple vs. Advanced Searching

---

The 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, as shown in [Figure 5.1](#). Protégé searches using the Preferred Name with exact matching.



Figure 5.1 Simple Search field and button

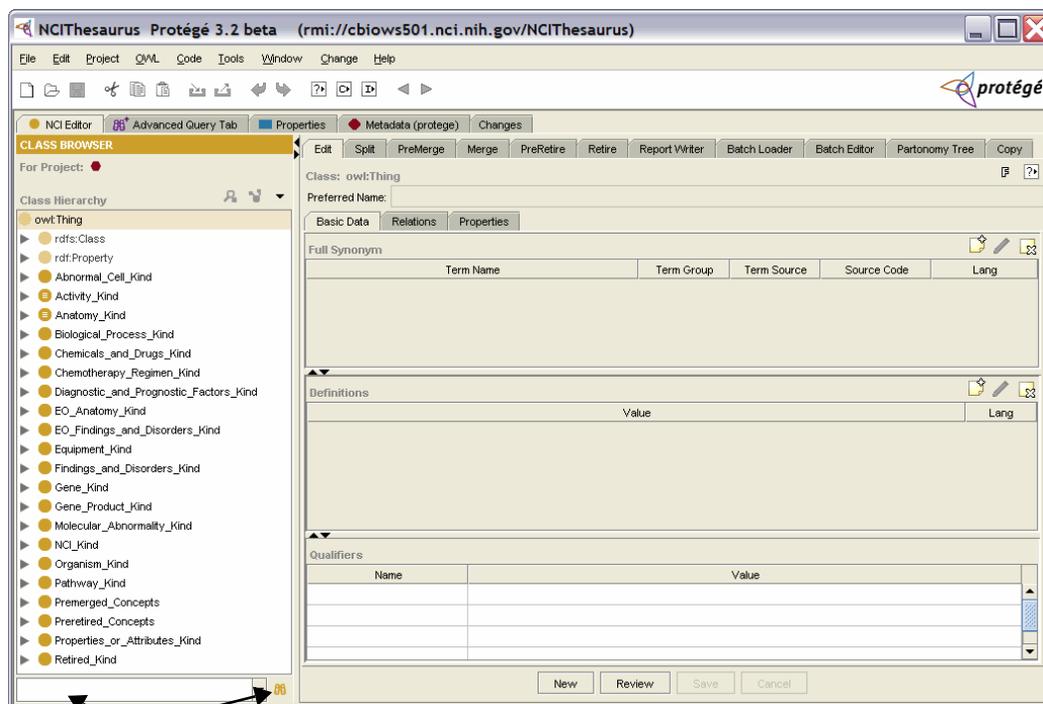
**Note:** For more information, see [Performing Simple Searches](#) on page 68.

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

**Note:** For more information about using the Advanced Query tab, see [Building Advanced Queries](#) on page 71.

## Performing Simple Searches

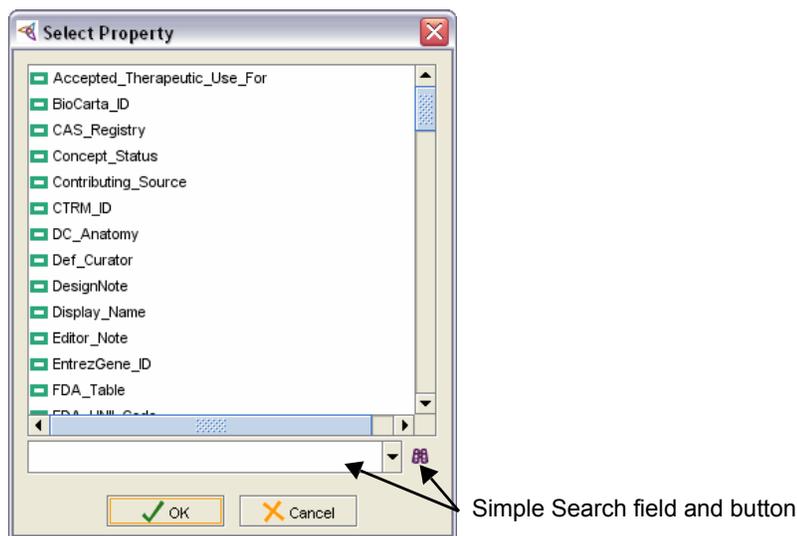
The **Simple Search** field and search button are available in multiple areas of the Protégé interface. When you first launch Protégé, you can find the field and button on the NCI Editor tab, just below the Class Browser, as shown in *Figure 5.2*.



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. As shown in *Figure 5.3*, the field and button are located below the list, in the bottom of the window.



Simple Search field and button

*Figure 5.3 Select Property window with Simple Search field and button*

## Using the Simple Search Field and Button

Follow these steps to perform a simple search:

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.



Type the search words here...

...then click here.

Figure 5.4 Simple Search field and Search for Class button

The Advanced search window opens, as shown in [Figure 5.5](#).

**Note:** The layout of this window is identical to the layout of the Advanced Query tab. (See [About the Advanced Query Tab](#) on page 60 for layout details.) For simple searches, you use only the features in the right side of the window to select, search for, and display results. For more information about the query-building features in the left side of the window, see [Building Advanced Queries](#) on page 71.

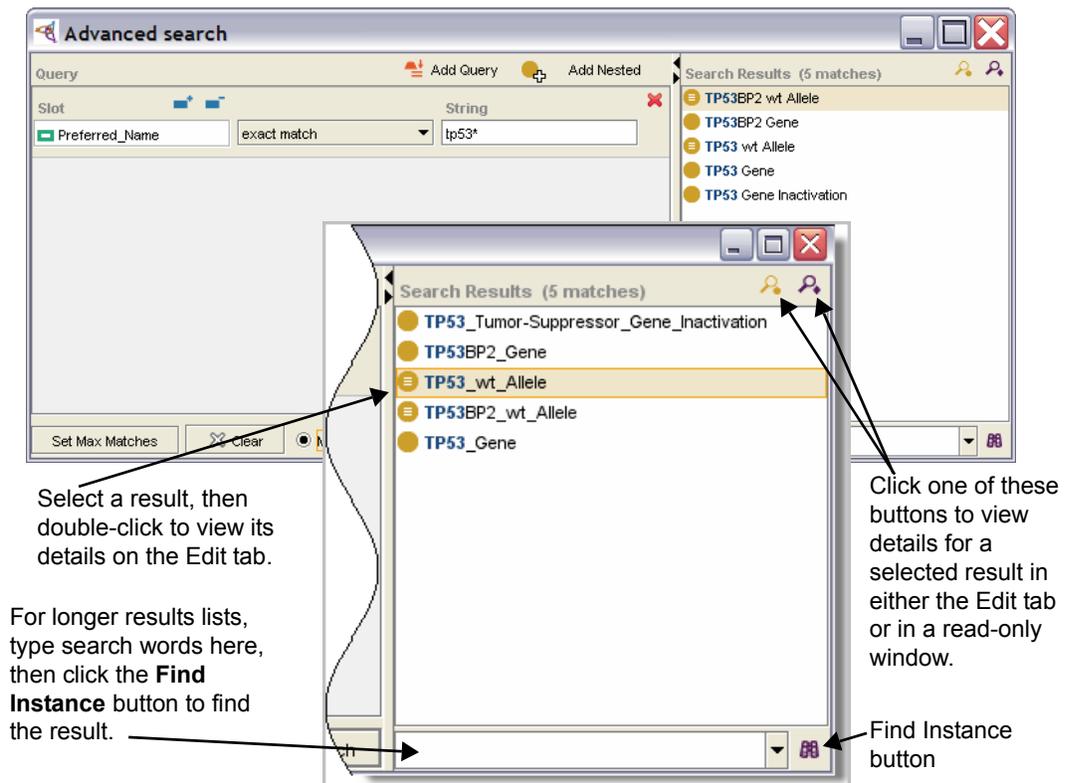


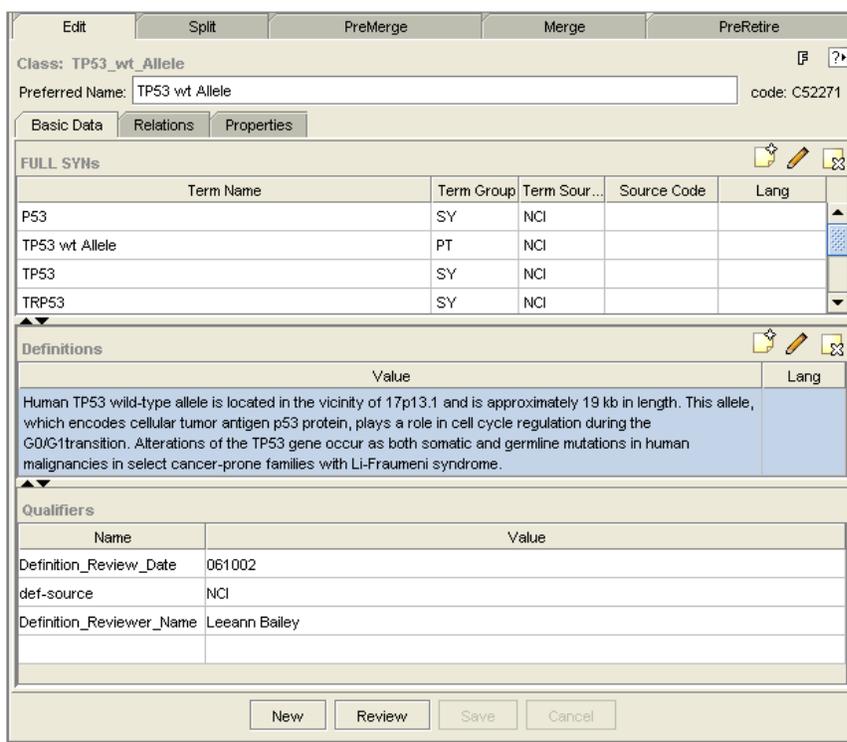
Figure 5.5 Results for simple search

## 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) Search within a long results list by doing the following:
  - a. Type a search word or phrase in the **Search** field below the list. (See [Figure 5.5](#) on the previous page.)
  - b. Click the **Find Instance** button .
2. Select a result, then do one of the following:
  - Double-click the selected result to view its basic data, relations, and properties in the **Edit** tab, as shown in [Figure 5.6](#). You may need to minimize or close the Advanced search window to see the information.
  - Click the **Edit Class** button  to view the basic data, relations, and properties for the selected result in the **Edit** tab, as shown in [Figure 5.6](#). You may need to minimize or close the Advanced search window to see the information.
  - 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.

[Figure 5.6](#) shows the result of the first two options. The Edit tab displays basic data for the selected result.



Class: TP53\_wt\_Allele [?] [x]

Preferred Name: TP53 wt Allele code: C52271

Basic Data **Relations** Properties

FULL SYNs [+] [x]

Term Name	Term Group	Term Sour...	Source Code	Lang
P53	SY	NCI		
TP53 wt Allele	PT	NCI		
TP53	SY	NCI		
TRP53	SY	NCI		

Definitions [+] [x]

Value	Lang
Human TP53 wild-type allele is located in the vicinity of 17p13.1 and is approximately 19 kb in length. This allele, which encodes cellular tumor antigen p53 protein, plays a role in cell cycle regulation during the G0/G1 transition. Alterations of the TP53 gene occur as both somatic and germline mutations in human malignancies in select cancer-prone families with Li-Fraumeni syndrome.	

Qualifiers

Name	Value
Definition_Review_Date	061002
def-source	NCI
Definition_Reviewer_Name	Leeann Bailey

New Review Save Cancel

*Figure 5.6 Edit tab with Basic Data for a selected result*

---

**Note:** The Advanced search window remains open unless you forcibly close it.

---

## Building Advanced Queries

Advanced queries give you more control over the scope and complexity of searches:

- You can search using a class name or any property, including a FULL\_SYN, definition, or preferred name.
- You can search for classes that have specific restrictions.
- You can build search queries using several search parameters, such as *contains*, *starts with*, or *ends with*.
- You can narrow or broaden the search scope by selecting a **Match All** (AND) or **Match Any** (OR) option.

The Advanced Query tab, shown in [Figure 5.7](#), enables you to build complex queries. For more information about the tab layout, see [About the Advanced Query Tab](#) on page 60. [Table 4.15](#) in that section (page 61) provides an alphabetical listing of the features.

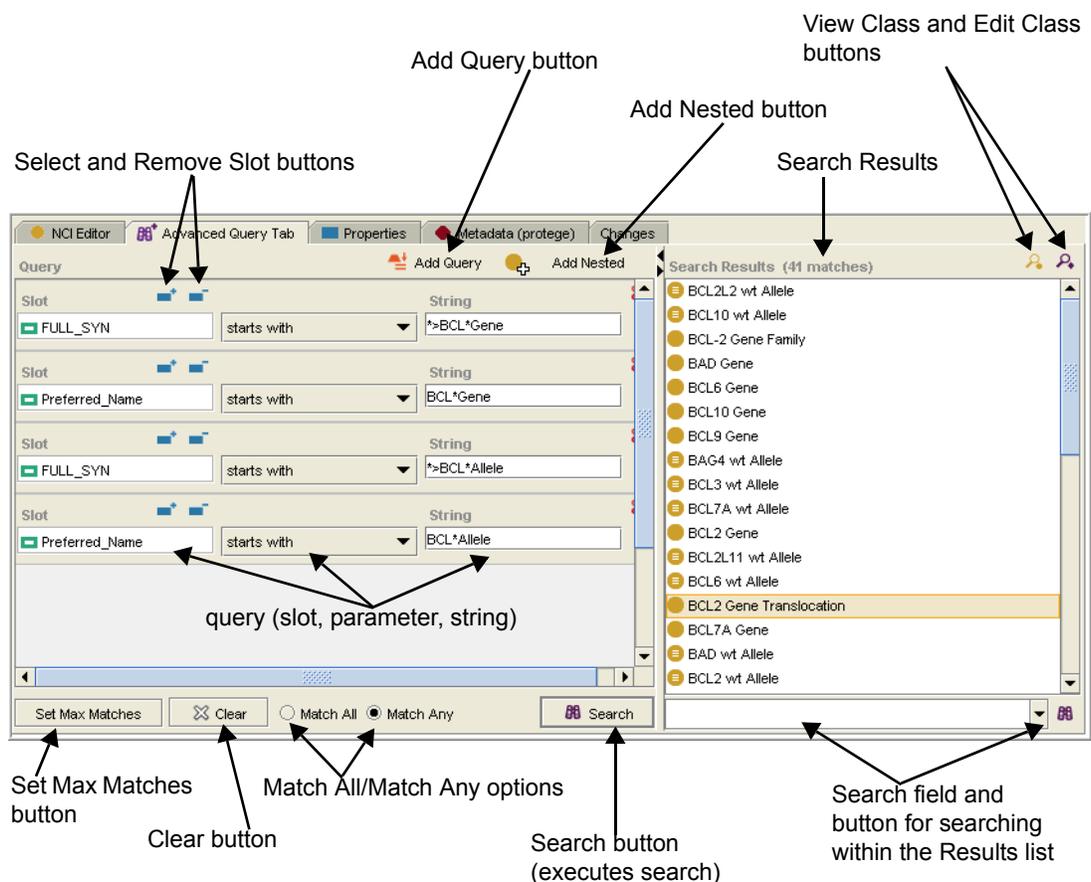


Figure 5.7 Advanced Query tab

## 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

- Build queries in the left pane of the Advanced Query tab. (See [Figure 5.7](#) on page 71.) 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.

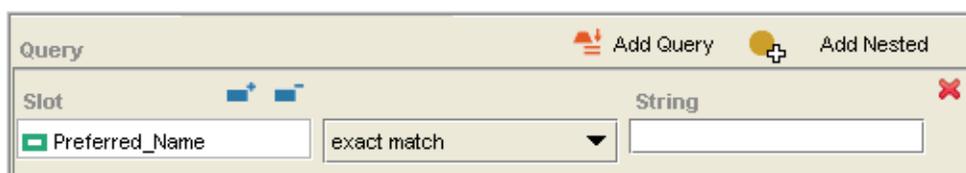


Figure 5.8 Default query

- 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.

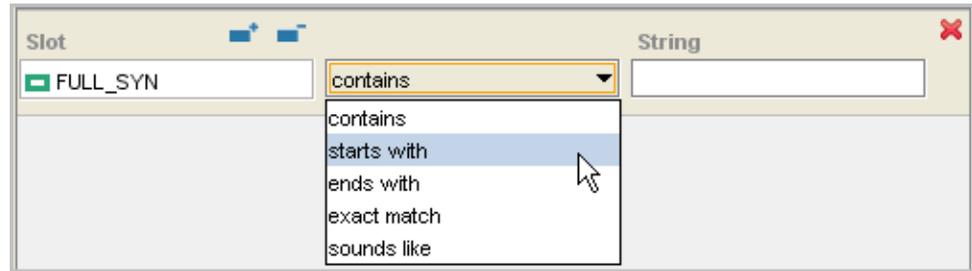
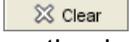


Figure 5.9 Parameter list

- Type a search string in the **String** field.
- Add queries using the **Add Query** button .
- Use the **Add Nested** button  when searching for restrictions or associations. For more information, see [Building a Restriction-Based Query](#) on page 81.
- To remove a single query, click the **Remove Query** button . If you have only one query and click this button, Protégé removes the query and displays the default query (*Preferred\_Name* and *exact match*).
- 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 .

## From here...

The rest of this section details procedures for building queries of varying complexity. It includes the following topics:

- [Building a Simple Property Query: Preferred Name](#) on page 74
- [Building a Complex Property Query: FULL\\_SYN](#) on page 75
- [Building a Combined Property Query: Preferred\\_Name and FULL\\_SYN](#) on page 77
- [Building a Restriction-Based Query](#) on page 81

## Building a Simple Property Query: Preferred Name

When you first open the Advanced Query tab, the left pane shows the default query of *Preferred\_Name exact match...* This example 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 decide to search using only the word *fluid*.

**Procedure:** Follow these steps to build and execute this query:

1. Click the **Advanced Query** tab.

As shown in [Figure 5.10](#), the Query pane on the left shows the default query: Preferred\_Name with exact matching. The **String** field is blank.

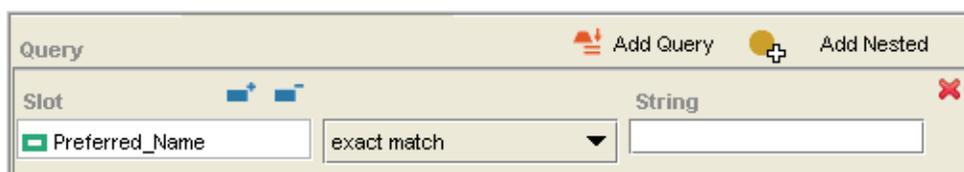


Figure 5.10 Default query

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 . This clears the **String** field and restores the default query, *Preferred\_Name exact match...*

5. Leave the search scope set to the default of **Match ALL** (Boolean AND).
6. Click the **Search** button .
7. In the Search Results pane on the right, double-click *Cerebrospinal\_Fluid*. Protégé switches to the **Edit** tab, which now displays FULL\_SYNs, definitions, and qualifiers for the selected class.

## Building a Complex Property Query: FULL\_SYN

As noted in the introduction to this section, queries using FULL\_SYNs require the opening (<) and/or closing (>) angle 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:** Follow these steps to build and execute this query:

1. Click the **Advanced Query** tab.

As shown in [Figure 5.11](#), the Query pane on the left shows the default query: Preferred\_Name with exact matching. The **String** field is blank.

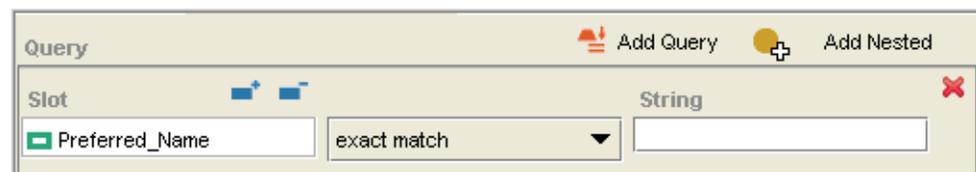


Figure 5.11 Default query

You now need to replace the default property with a FULL\_SYN property.

2. Click the **Select Slot** button .
3. In the Select Slot window (shown in [Figure 5.12](#)),
  - scroll until you see FULL\_SYN property, or
  - use the **Search** field and button to search for it.

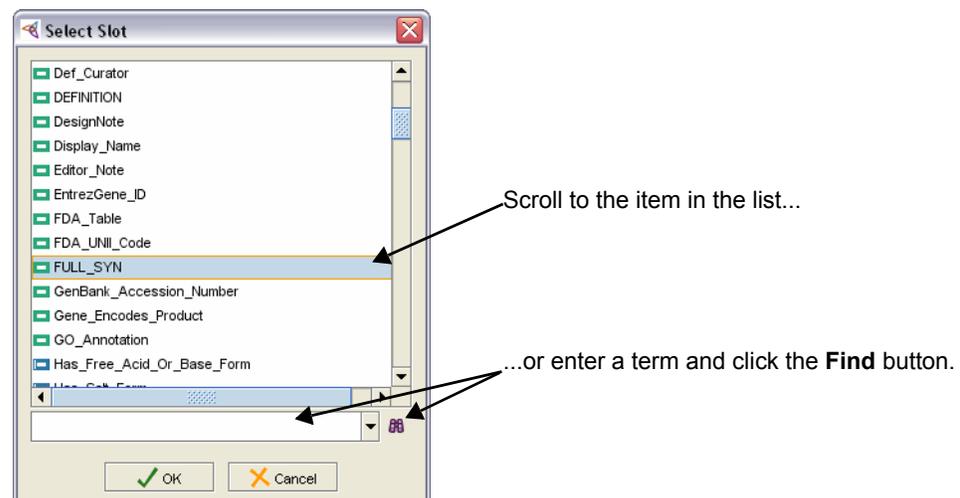


Figure 5.12 Select Slot window

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

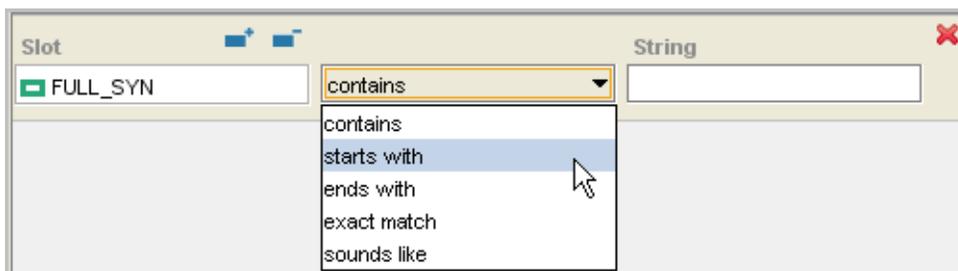


Figure 5.13 Parameter list - starts with

6. In the **String** field, type **\*>benign**.  
The full query is complete.

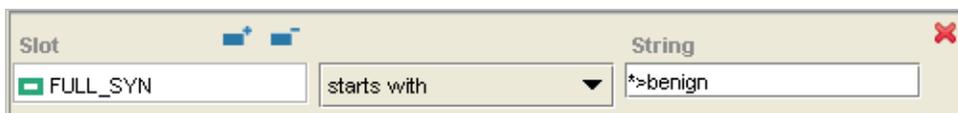


Figure 5.14 Finished query

7. Leave the search scope set to the default of **Match All** (Boolean AND).
8. Click the **Search** button  .
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 **Edit** tab, which now displays FULL\_SYNS, definitions, and qualifiers for the selected class.

## Building a Combined Property Query: Preferred\_Name and FULL\_SYN

Previous examples in this chapter have used only one query. 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:** Follow these steps to build and execute this query:

1. Click the **Advanced Query** tab.

As shown in [Figure 5.15](#), the Query pane on the left shows the default query: Preferred\_Name with exact matching. The **String** field is blank.

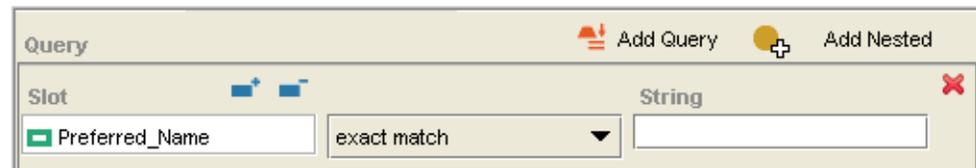


Figure 5.15 Default query

This 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.

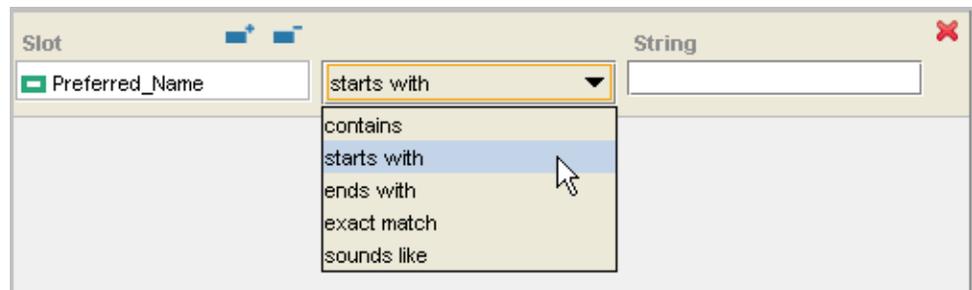


Figure 5.16 Selecting from the Parameter list

3. In the **String** field, type *bc1\*gene*, as shown in [Figure 5.17](#). Since you are using the Preferred\_Name property for this query, you can omit the angle brackets.

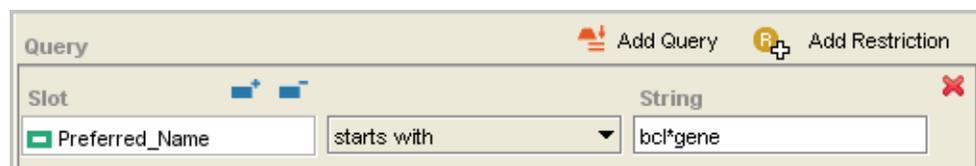


Figure 5.17 First completed query

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

4. Click the **Add Query**  button.  
The new query shows the default query using Preferred\_Name as the slot. You now need to change the slot to FULL\_SYN.
5. Click the **Select Slot** button .
6. In the Select Slot window (shown in [Figure 5.18](#)),
  - scroll until you see FULL\_SYN property, or
  - use the **Search** field and button to search for it.

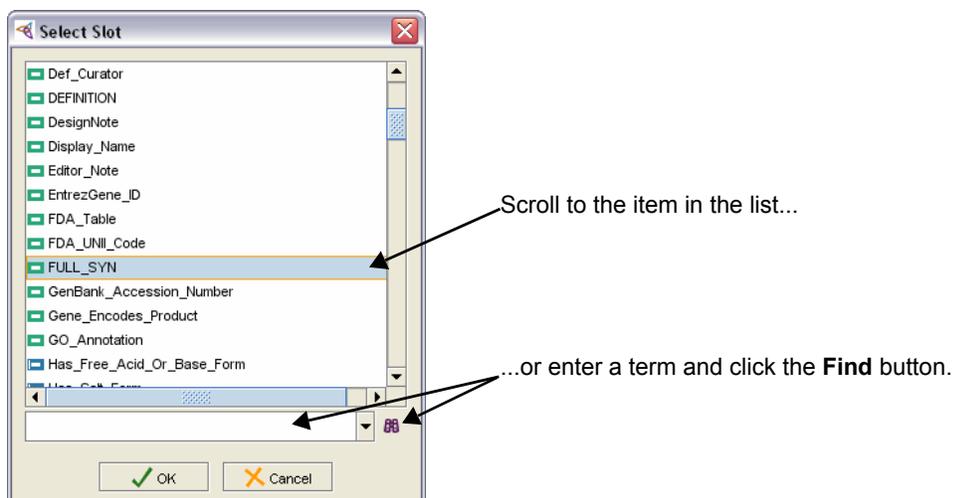


Figure 5.18 Select Slot window

7. Click **OK** to close the window.
8. Change the search parameter to **starts with**.
9. In the **String** field, type *\*>BCL\*Gene*, as shown in [Figure 5.19](#). Since FULL\_SYN is a complex property, include angle brackets.



Figure 5.19 Second completed query

Two more queries are required:

- *Preferred\_Name starts with bcl\*allele*
  - *FULL\_SYN starts with \*>bcl\*allele*
10. Click the **Add Query**  button.
  11. Leave **Preferred\_Name** as the slot.
  12. Change the search parameter to **starts with**.
  13. In the **String** field, type *bcl\*allele*.
  14. Click the **Add Query**  button.

15. Change the slot to **FULL\_SYN**.
16. Change the search parameter to **starts with**.
17. In the String field, type **\*>bcl\*allele**. The combined queries should now resemble *Figure 5.20*.

*Figure 5.20 Combined queries*

18. Change the scope **Match Any** (Boolean OR).
19. Click the **Search** button Search. *Figure 5.21* shows the full query and the accompanying Search Results list.

*Figure 5.21 Full query with search results*

20. Do either of the following:
  - Double-click a result to view its details on the **Edit** tab.
  - Click the **Clear** button to clear the query pane and revise your search.

Try the following variations, noting the results for each:

1. Use **Match Any** for the following queries:
  - Single query: *FULL\_SYN starts with \*>bcl*
  - Two queries:
    - *Preferred\_Name contains bcl*
    - *FULL\_SYN contains bcl*
2. Use **Match All** for these queries:
  - Single query: *FULL\_SYN contains bcl*
  - Single query: *FULL\_SYN contains translocation*

## Building a Restriction-Based Query

As noted in the introduction to this section, the Advanced Query tab enables you to search for classes that have specific restrictions.

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

**Procedure:** Follow these steps to build and execute this query:

1. Click the **Advanced Query** tab.

2. Click the **Add Nested** button  **Add Nested**.

Protégé places the restriction below the default query (Preferred\_Name with exact matching). You don't need the default query, so in the next step, you'll remove it.

3. Delete the default query by clicking the **Remove Query** button , as shown in [Figure 5.22](#).

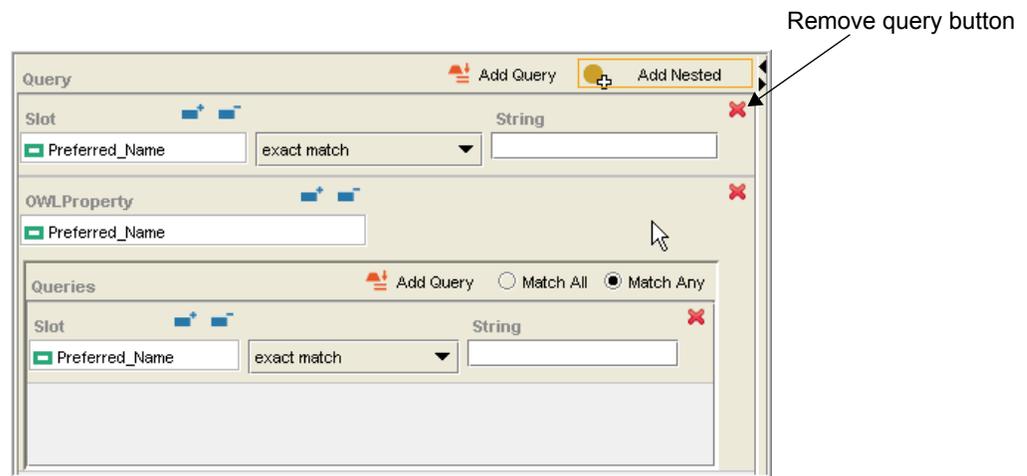


Figure 5.22 Removing the default query

The left side of the Advanced Query tab should now resemble [Figure 5.23](#).

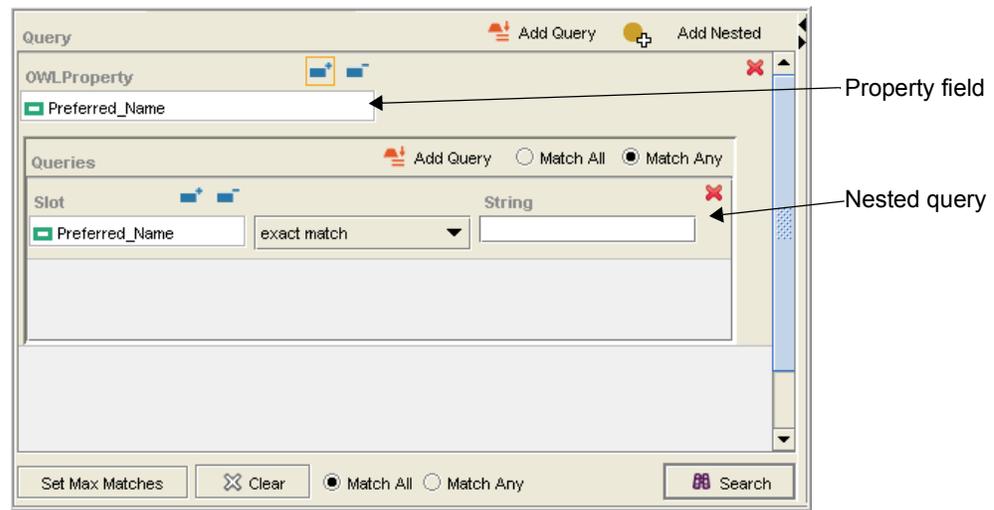


Figure 5.23 Initial setup for restriction-based query

In the next step, you'll change the upper Slot field so that it shows the property *Gene\_Plays\_Role\_In\_Process*.

4. Click the **Select Slot** button .
5. In the Select Property window, select or search for the following property:  
*Gene\_Plays\_Role\_In\_Process*

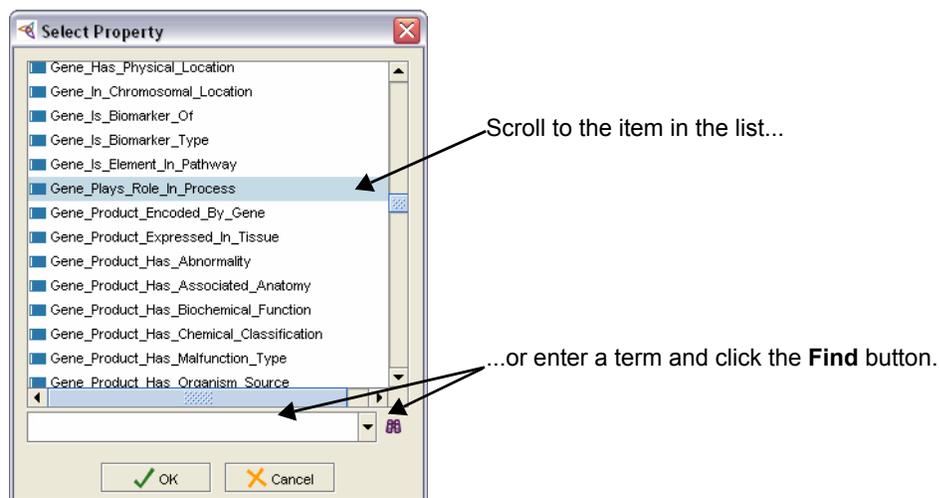


Figure 5.24 Select Property window

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

The upper Slot field now shows *Gene\_Plays\_Role\_In\_Process*, as shown in [Figure 5.25](#).

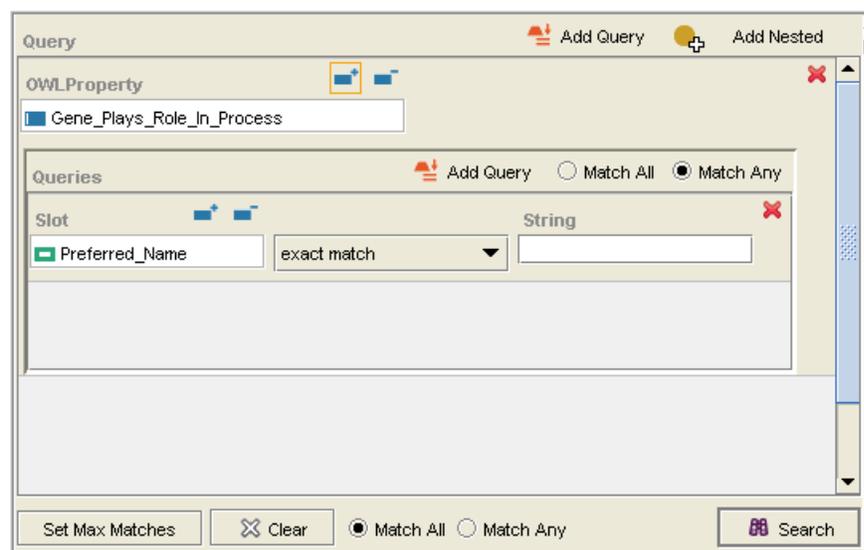


Figure 5.25 Restriction-based query with search property

7. In the Queries area, 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 5.26](#).

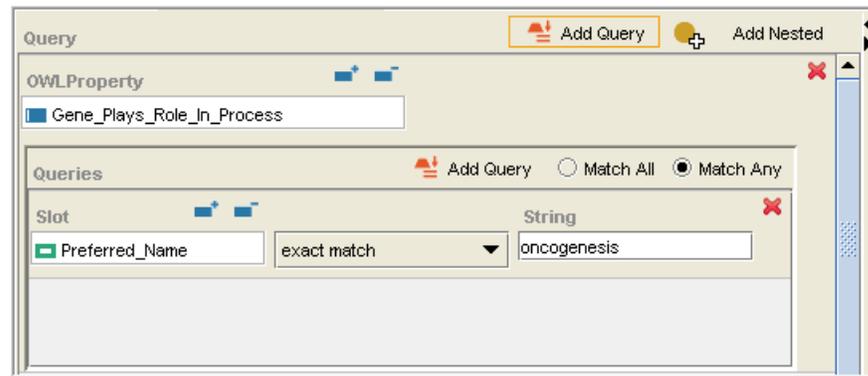


Figure 5.26 Restriction-based query with completed fields

10. Click the **Search** button . The results are displayed on the right, as shown in [Figure 5.27](#).

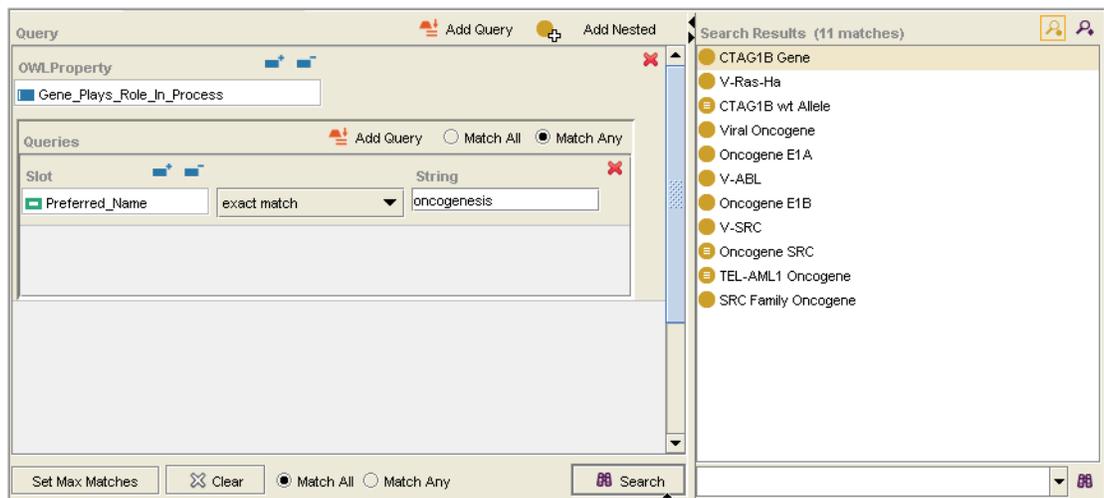


Figure 5.27 Search results for restriction-based query

11. Select a result, then do one of the following:
- Double-click the selected result to view its basic data, relations, and properties in the **Edit** tab, as shown in [Figure 5.6](#). Protégé switches to the Edit tab.
  - Click the **View Class** button  to view the basic data, relations, and properties for the selected result in the **Edit** tab, as shown in [Figure 5.6](#). You may need to minimize or close the Advanced search window to see the information.
  - 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.

## Configuring the Advanced Query Tab

As mentioned previously, when you initiate a simple search, Protégé uses the Preferred Name with exact matching. The default query on the Advanced Query tab illustrates this convention, as shown in [Figure 5.28](#).

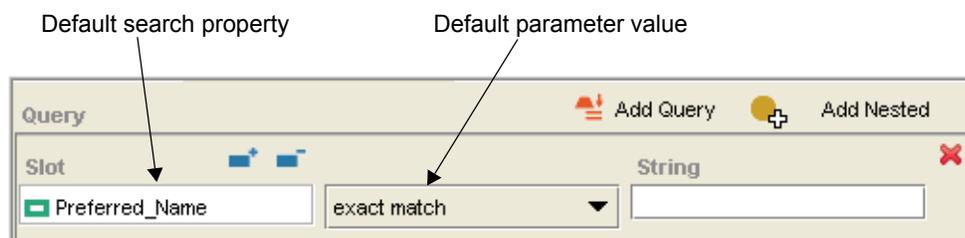


Figure 5.28 Default query - Preferred Name with exact matching

You can customize Protégé to use a different default search property (or *Slot*) and search parameter value. Configuration information is stored in the *protege.properties* file, which appears in *C:\Program Files\Protégé\_3.2\_beta*. Fortunately, you can customize your configuration from a window in Protégé. You are not required to edit the file stored in the application directory.

The following section explains how to change both the default search property and the default search parameter. The example used in the procedures shows you how to make the following changes:

- Change the default search property to *FULL\_SYN*.
- Change the default search parameter to *contains*.

Once you complete the procedures, the default query on the Advanced Query tab should resemble [Figure 5.29](#).

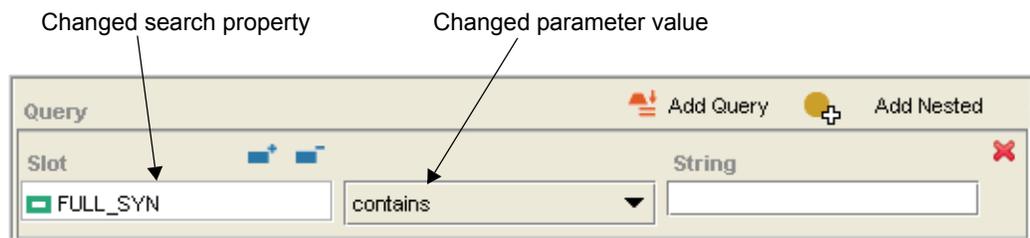


Figure 5.29 Default query with new search property and search parameter

## Changing the Default Search Property

To change the default search parameter to *FULL\_SYN*, follow these steps:

1. Select the following menu command: **Project > Configure**.
2. In the Configure file window, click the **Property Files** tab.
3. Ensure that the **protege.properties** tab is selected, as shown in [Figure 5.30](#). The two columns in this tab are labeled **Property** and **Value**. 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.

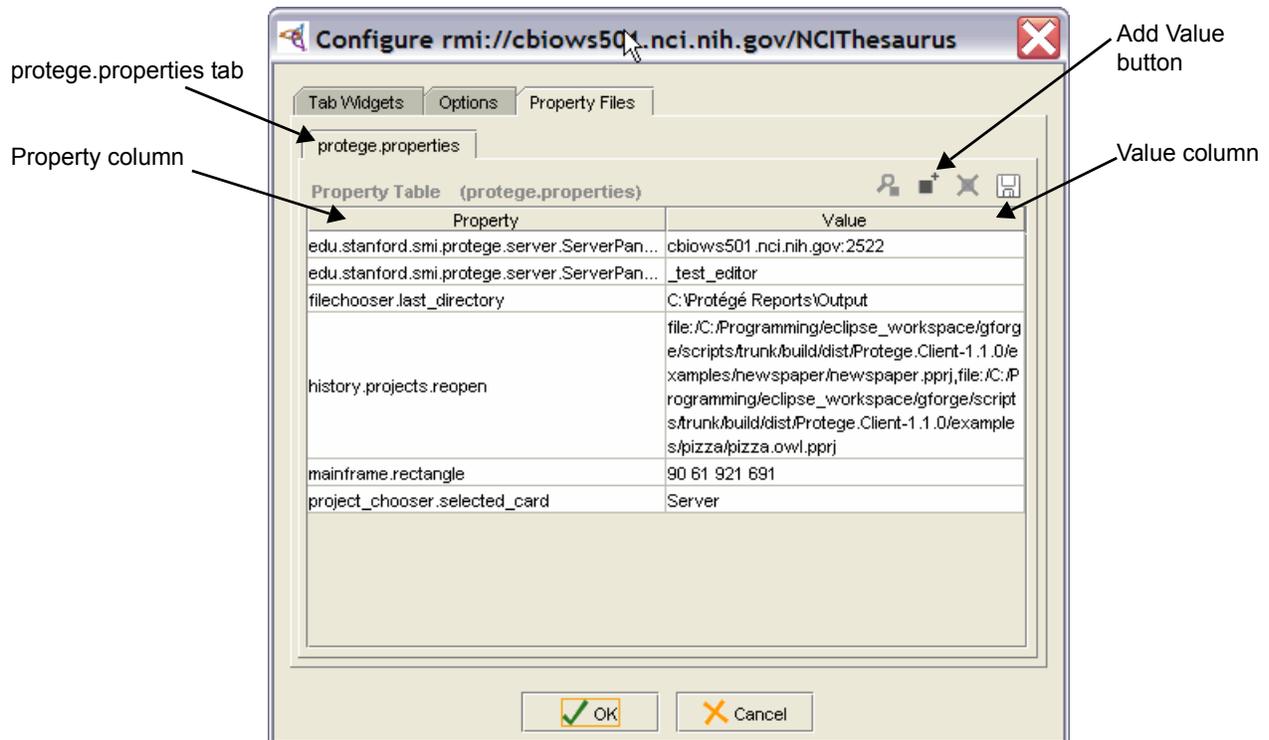


Figure 5.30 Configure file window with *protege.properties* tab selected

4. On the upper right, click the **Add Value** button  (second from left). An editable text field appears in the **Property** column, as shown in [Figure 5.31](#).

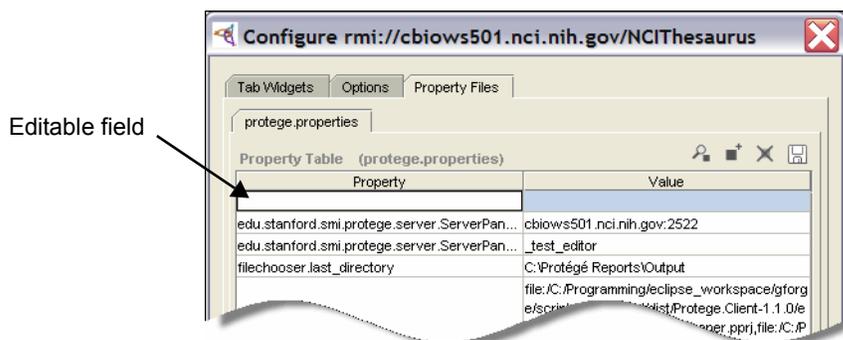


Figure 5.31 Configure file window - *Property Files* tab with editable field

- In the **Property** field, type the following exactly as shown:

*query\_plugin.default.search\_slot*

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

- 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 moves the new property-value combination to the bottom of the list.
- In the **Value** field, type *FULL\_SYN*.
- Press TAB to move the cursor out of the field.

---

**Note:** Leave the Configure file window open for the next section, in which you will change the default search parameter to *contains*.

---

### Changing the Default Search Parameter

- With the Configure file window still open, click the **Add Value** button  to add another name-value pair. A new, editable field appears in the Property column on the left.
- In the **Property** field, type the following exactly as shown:  
*query\_plugin.default.search\_type.String*
- Press TAB to enable the empty field in the **Value** column on the right. This step activates the **Value** field and moves the new property-value combination to the bottom of the list.
- In the **Value** field, type *contains*. The Configure file window should now resemble [Figure 5.32](#).

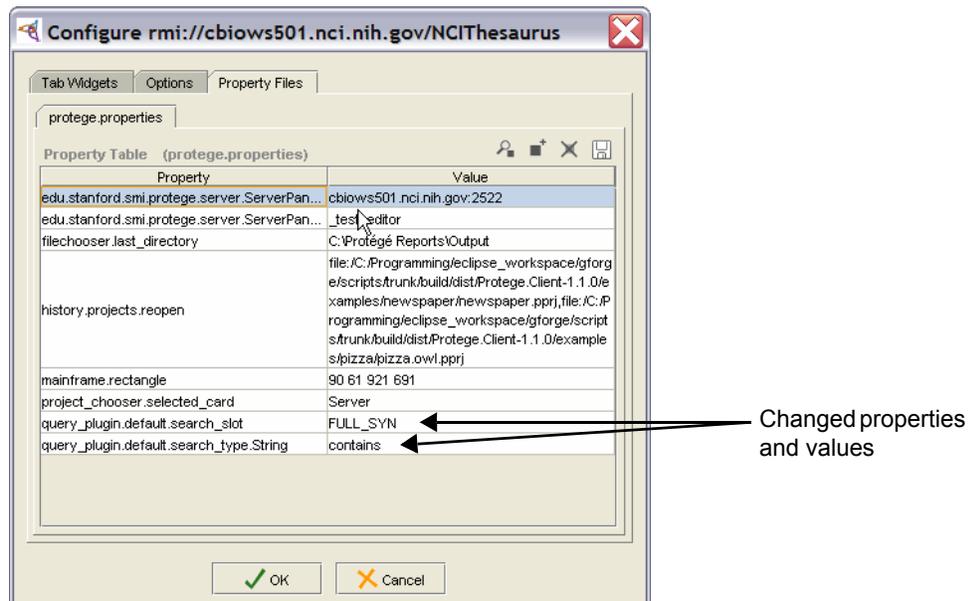


Figure 5.32 Configure file window with new search property and parameter

- Click **OK** to close the Configure file window.

## Verifying the New Search Configuration

The best way to verify that the new configuration is in effect is to check the default query on the Advanced 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** tab.
2. Select a class in the Class Browser on the left, ensuring that you select a different class from the one that is currently displayed. This step refreshes the interface.
3. Click the **Advanced 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.

---

## Search Properties and Values in the `protege.properties` File

As discussed on page 84, configuration information is stored in the `protege.properties` file, which appears in `C:\Program Files\Protégé_3.2_beta`. The default configuration is represented in that file as

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

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

If you completed the steps on pages page 85 and page 86, 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 sections use strings. Other types may be used at the NCI in the future, and they are listed here for your information.

*Table 5.1* lists the data types implemented by SMI and shows the property name and accepted values for each.

**Note:** In the Possible Values column, *is* means *equals*.

<b>Data Type</b>	<b>Property Name</b>	<b>Possible Values</b>
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.CIs	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 5.1 Search data types implemented by SMI*

# CHAPTER 6

## USING BASIC EDITING FEATURES

This chapter explains how to use the NCI Protégé Plug-In to perform core editing procedures. It includes the following topics:

- *About Classes* on this page
- *Creating a Class* on page 94
- *Treeing Classes* on page 98
- *Asserting Annotation Properties* on page 103
- *Asserting Relations* on page 107
- *Adding an Association* on page 119

The procedures in this chapter are all performed using the **Edit** tab and its three subtabs: **Basic Data**, **Relations**, and **Properties**. For more information about the layout of the tab and subtabs, see the following sections in *Chapter 4*:

- *Edit Tab* on page 45
- *Basic Data Subtab* on page 46
- *Relations Subtab* on page 47
- *Properties Subtab* on page 48

### 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 used to identify classes. Each class within the Thesaurus must therefore be uniquely named. 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.

To support OWL compliance, underscores and dashes are the only punctuation that can be used in class names. 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. Unlike a class name, the preferred name can be changed. Both class names and preferred names are generally singular.

[Figure 6.1](#) shows the name, preferred name, and code for the class *CTAG1\_wt\_Allele*, as displayed on the Basic Data subtab of the Edit tab.

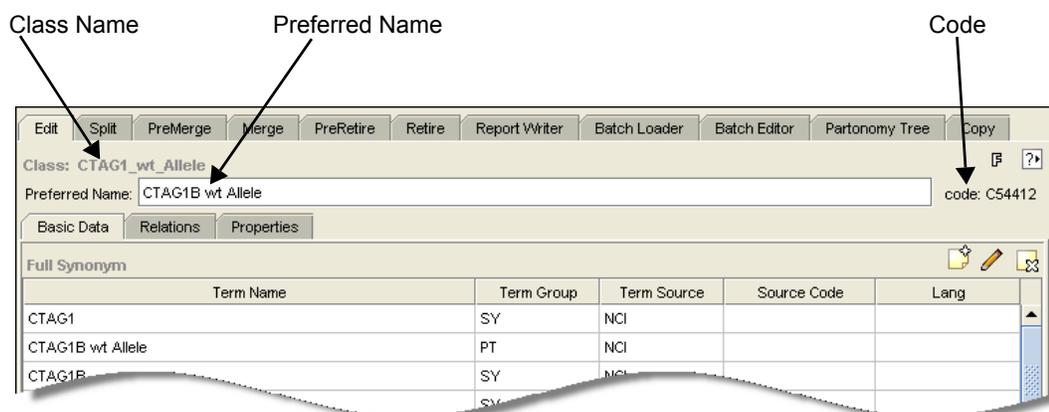


Figure 6.1 Class information displayed on Edit tab - Basic Data subtab

## Class Relationships

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

Class relationships are displayed in the Class Browser pane of the NCI Editor tab. For more information, see [Figure 4.2](#) on page 43.

## Class Properties

*Properties* are used to 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 next sections 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*, and *Definition\_Reviewer\_Name*. As shown in [Figure 6.2](#), the full definition and qualifiers are shown on the Basic Data subtab of the Edit tab.

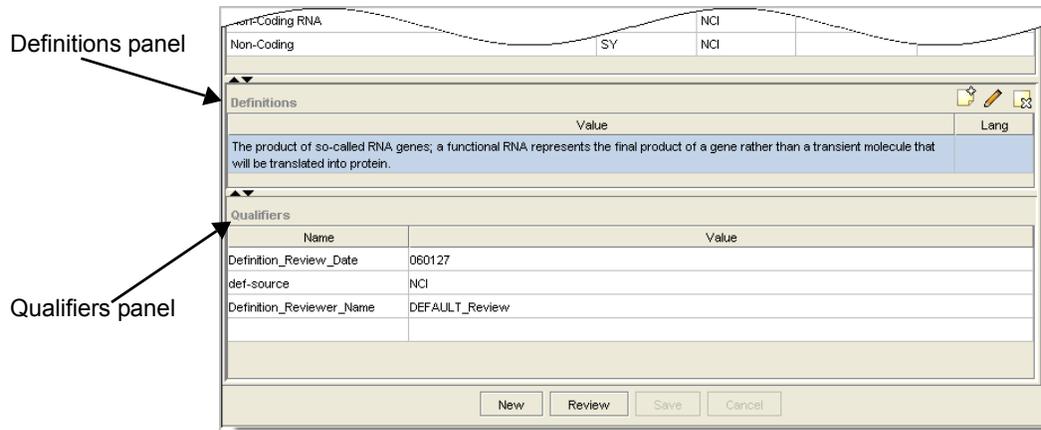


Figure 6.2 Definitions and Qualifiers panels

The goal for NCI editors is 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.

**Tip:** You can compose or edit a definition in a word processor, check the spelling, then copy and paste the definition into the Protégé editing window.

When writing or editing a definition, add the source attribution to the *attr* field in the Edit DEFINITION Annotation Property window. Observe the following rules.

**For definitions quoted from sources:**

- 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*.

**For 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.

**For definitions using journal articles as sources:**

Cite the source as follows:

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

**For 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](http://www.lef.org/prod_hp/abstracts/biostimabs.html).

**For 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.

---

**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 *Pre-Merge: Flagging Classes to be Merged* on page 127 and *Pre-Retire: Flagging a Class for Retirement* on page 131.

---

## Reviewing Changes As You Work

As you work with Protégé, you will often use the three subtabs of the **Edit** tab (Basic Data, Relations, and Properties) to view and edit class data. You can freely switch between those subtabs without having to click the **Save** button each time. Protégé holds your changes in memory as long as you continue working with data shown on the Edit tab for the currently focused class.

**Caution:** If you switch to another parent tab (such as Split, PreMerge, or Report Writer) without saving your changes, a Confirmation window prompts you to save. Your changes are discarded unless you answer **Yes** to the prompt.

Before saving changed data, you can view a data summary in a convenient Review window that summarizes the data for the current class, including changes that you have made but have not yet saved.

To open the Review window, click the **Review** button. It is the second of the four buttons at the bottom of the **Edit** tab:

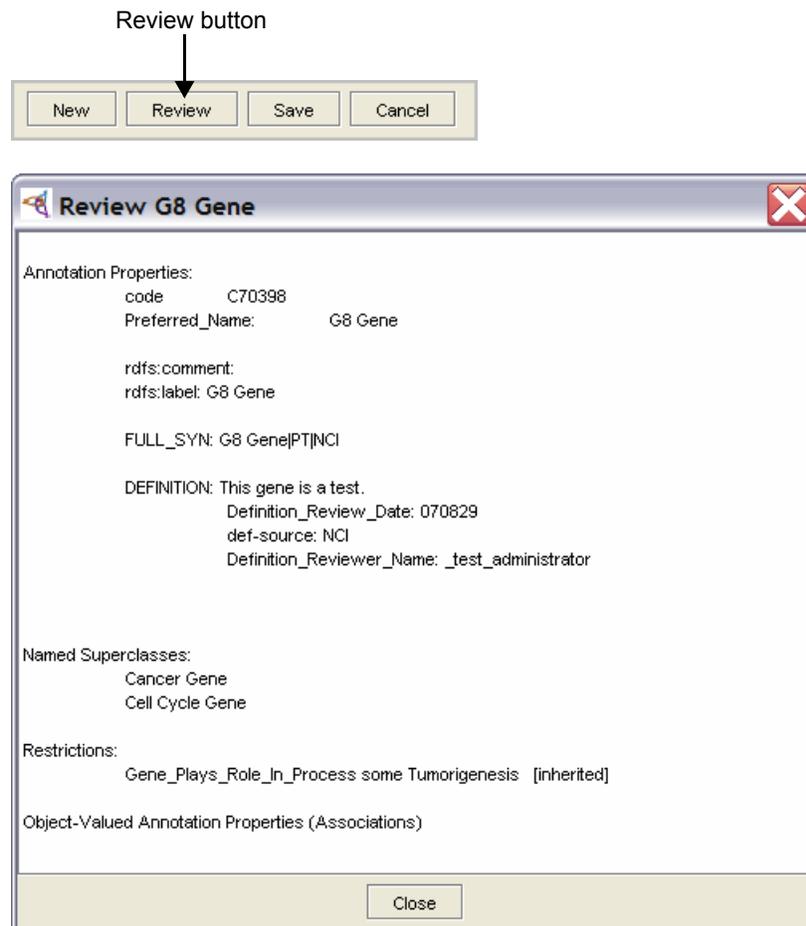


Figure 6.3 Review window

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

## Creating a Class

Before creating a new class, search thoroughly to find 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.

**Note:** For more information about searching in Protégé, see *Searching: Simple and Advanced* on page 67.

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. Select any class in the Class Browser on the left (except *owl:Thing*).

**Note:** The class you select on the left doesn't have to be the parent class for the new class. You are performing this step because you can't use the Edit tab unless something other than *owl:Thing* is selected in the Class Browser.

2. Click the **Edit** tab on the right if it is not already displayed.
3. Click the **Basic Data** subtab.
4. Click the **New** button at the bottom of the tab, as shown in *Figure 6.4*.

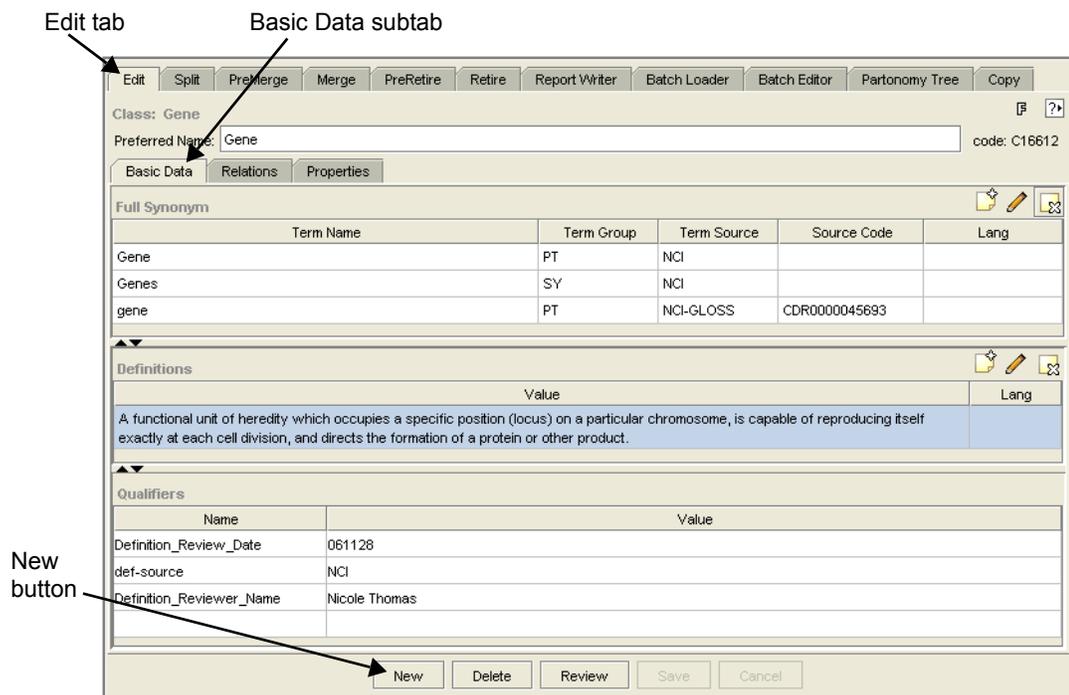


Figure 6.4 Creating a new class

5. In the Create Subclass window, click the **Select Superclass** button , shown in [Figure 6.5](#). This step enables you to select a parent class.

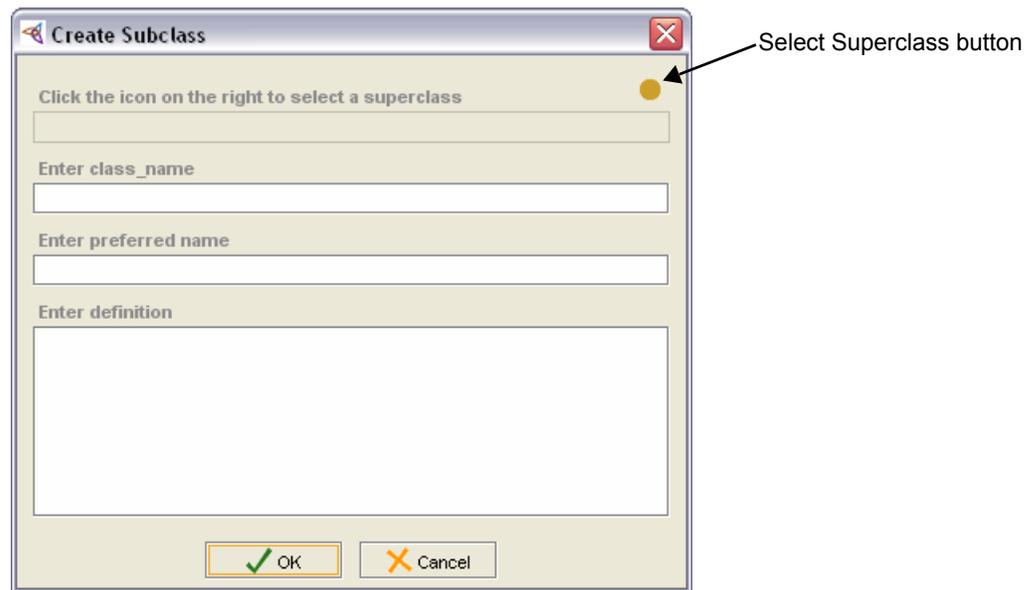


Figure 6.5 Create Subclass window

6. In the Select a superclass window (shown in [Figure 6.6](#)),
- select a class from the list, or
  - type a value in the **Search** field and click the **Search for Class** button.

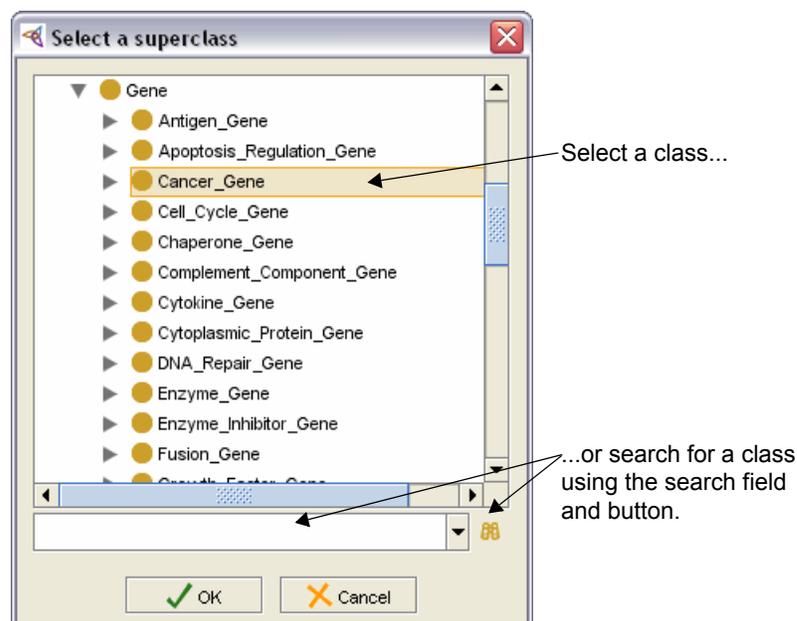


Figure 6.6 Select a superclass window

- (Search option only) When the Advanced Query window opens, select a search result, then click **OK** to close the window and return to the Select a superclass window.

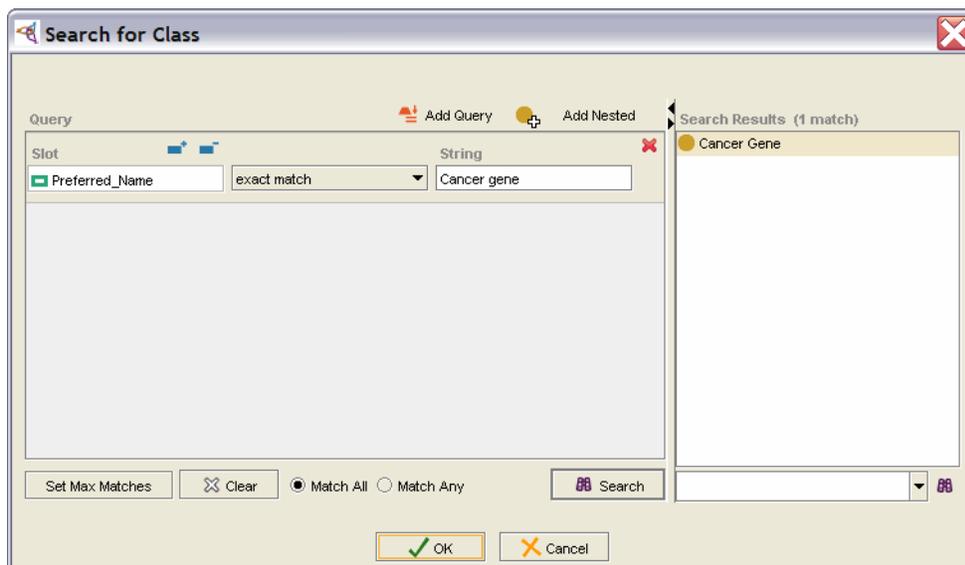


Figure 6.7 Advanced Query window - Search for Class

- 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, as shown in [Figure 6.8](#).
- Enter a class name, preferred name, and definition.
- (Optional) If you wrote the definition in a word processor, copy and paste the definition into the **Enter definition** field.
- Verify that the Create Subclass window now resembles [Figure 6.8](#).

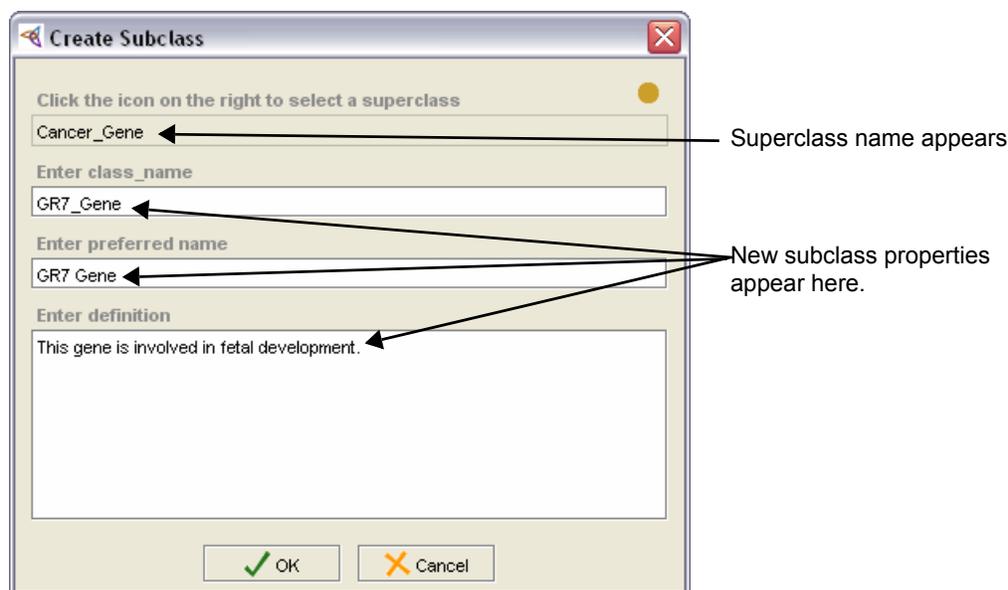


Figure 6.8 Create Subclass window with completed fields

12. Click **OK** to close the Create Subclass window.

13. Verify the following results:

- In the main Protégé window, the new class is now highlighted in the Class Hierarchy on the left.
- The **Basic Data** subtab (shown in [Figure 6.9](#)), shows the following information:
  - Full Synonym panel: Term Name, Term Group, and Term Source.
  - Definitions panel: shows the definition
  - Qualifiers panel: *Definition\_Review\_Date*, *def-source*, and *Definition\_Reviewer\_Name*.

The screenshot displays the Protégé interface. On the left, the 'CLASS BROWSER' shows a class hierarchy with 'G8 Gene' selected. On the right, the 'Basic Data' subtab is active for the 'G8\_Gene' class. The 'Preferred Name' is 'G8 Gene' and the 'code' is 'C70398'. The 'Full Synonym' panel contains a table:

Term Name	Term Group	Term Source	Source Code	Lang
G8 Gene	PT	NCI		

The 'Definitions' panel shows a single definition:

Value	Lang
This gene is involved in fetal development.	

The 'Qualifiers' panel shows a table:

Name	Value
Definition_Review_Date	070829
def-source	NCI
Definition_Reviewer_Name	_test_administrator

At the bottom of the window, there are buttons for 'New', 'Delete', 'Review', 'Save', and 'Cancel'.

Figure 6.9 New class information

## Treeing Classes

---

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.

---

### In This Section...

- [Adding a Parent Class](#) on this page
- [Modifying a Parent Class](#) on page 101
- [Deleting a Parent Class](#) on page 102

## Adding a Parent Class

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

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

1. Select the class to which you want to add a parent class in the Class Browser on the left.
2. Click the **Edit** tab 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 , as shown in [Figure 6.10](#)

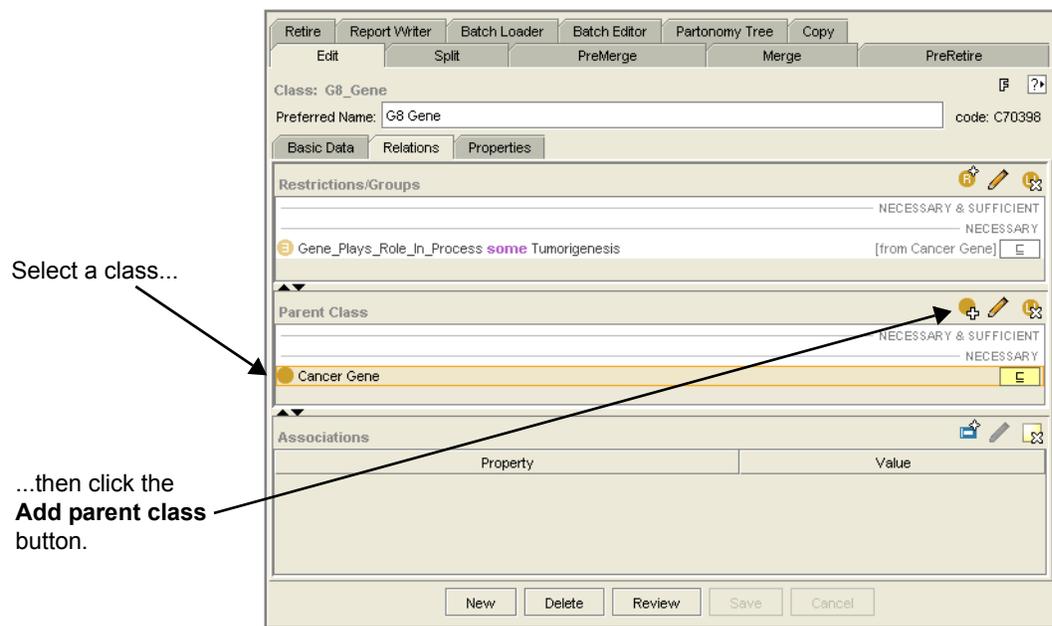


Figure 6.10 Edit Tab - Parent Class panel

6. In the Select a class window, (shown in [Figure 6.11](#)),
  - select a class from the list, or
  - type a value in the **Search** field and click the **Search** button.
7. (Search option only) When the Advanced Query window opens, select a search result, then click **OK** to close the window and return to the Select a class window.

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

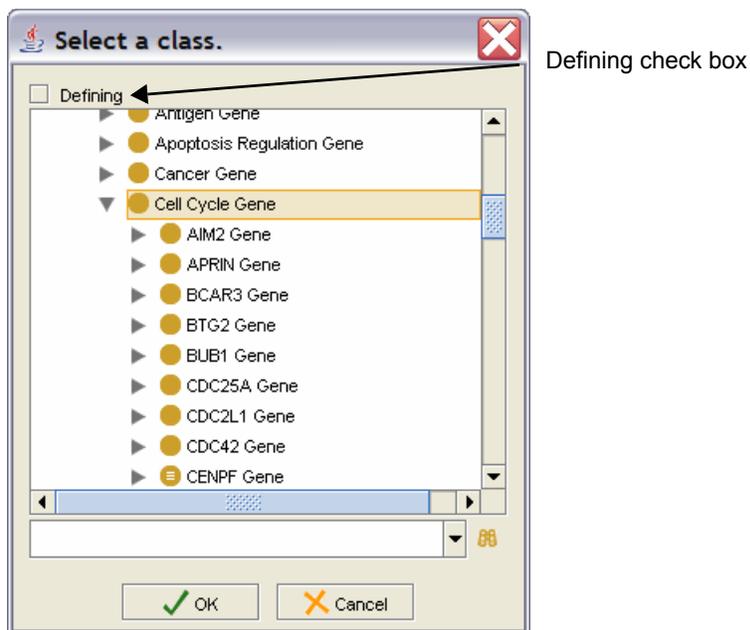


Figure 6.11 Select a class window - Defining check box

- Click **OK** to close the Select a class window. As shown in [Figure 6.12](#), the Parent Class panel shows the new class.



Figure 6.12 Parent Class panel with new class

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

- Click the **Save** button to accept the change, then close any confirmation messages that appear.

## Modifying a Parent Class

To modify a parent class, follow these steps:

1. Select the class to be modified in the Class Browser on the left.
2. Click the **Edit** tab on the right 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.
6. In the Modify Named Superclass window shown in [Figure 6.13](#), click the **Select a superclass** button on the right.

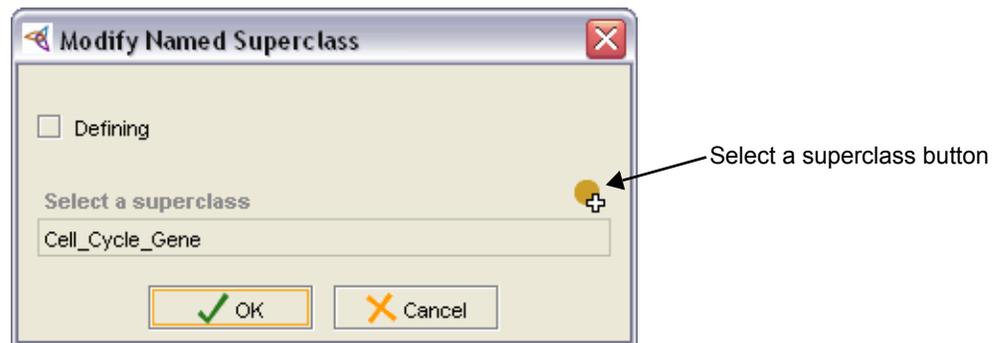


Figure 6.13 Modify Named Superclass window

7. In the Select a named class window (shown in [Figure 6.14](#)),
  - select a class from the list, or
  - type a value in the **Search** field and click the **Search** button.

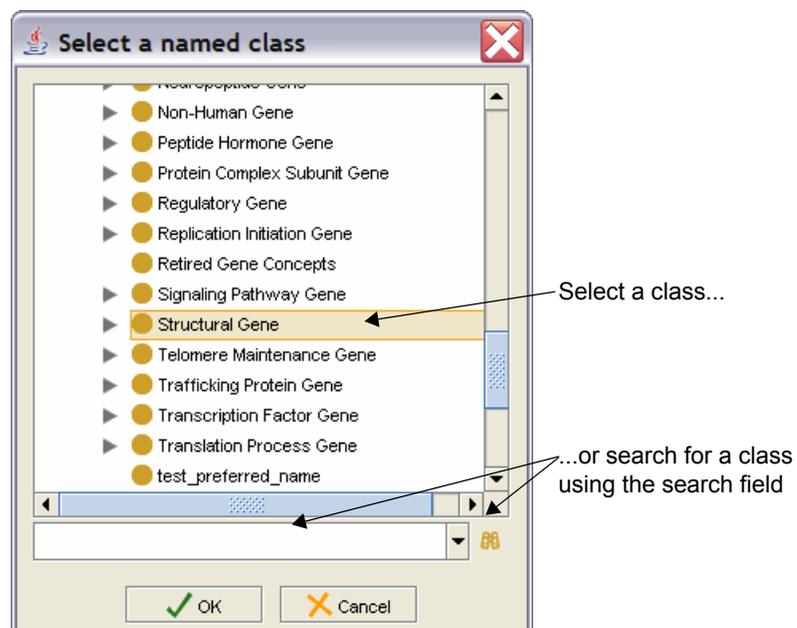
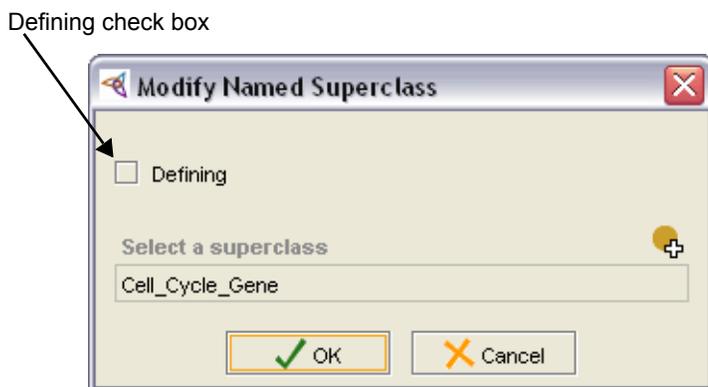


Figure 6.14 Select a named class window

8. (Search option only) When 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.
9. Click **OK** to close the Select a named class window.
 

**Note:** As shown in *Figure 6.15*, the original class name still appears in the Modify Named Superclass window. The new name will show when you return to the Parent Class panel.
10. (Optional) If the new class is a defining class, check the **Defining** box.



*Figure 6.15 Modified parent class*

11. Click **OK** to close the Modify Named Superclass window.
12. Click the **Save** button to accept the change, then close any confirmation messages that appear.

## Deleting a Parent Class

To delete a parent class, follow these steps:

1. In the Class Browser on the left, select the class for which you want to delete a parent class.
2. Click the **Edit** tab on the right 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, then close any confirmation messages that appear.

## Asserting Annotation Properties

---

### Tips for Working with Properties

- *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.

### In This Section...

- *Adding a Full Synonym* on page 104
- *Modifying a Full Synonym* on page 105
- *Deleting a Full Synonym* on page 105
- *Modifying a Definition* on page 106

## Adding a Full Synonym

To add a full synonym, follow these steps:

1. Select the class requiring a new synonym in the Class Browser on the left.
2. Click the **Edit** tab on the right 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.
5. In the Create FULL\_SYN Annotation Property window (shown in [Figure 6.16](#) on page 104), type 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.

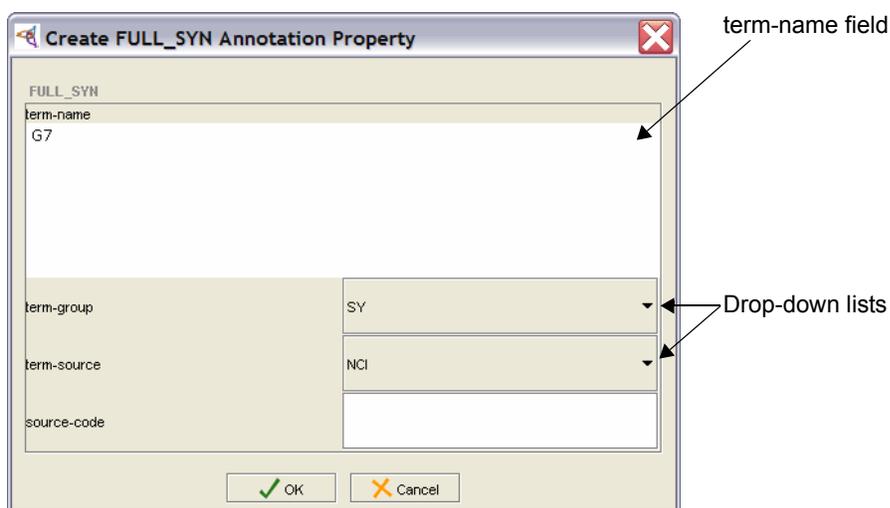


Figure 6.16 Create FULL\_SYN Annotation Property window

7. Click **OK** to close the window. The new synonym now appears in the Full Synonym list.

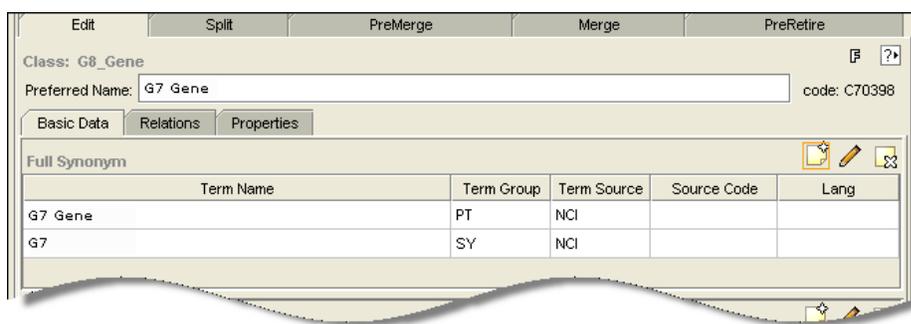


Figure 6.17 New synonym

8. Click the **Save** button to accept the change, then close any confirmation messages that appear.

## Modifying a Full Synonym

To modify a full synonym, follow these steps:

1. Select the class to be modified in the Class Browser on the left.
2. Click the **Edit** tab on the right if it is not already displayed.
3. Click the **Basic Data** subtab to view basic properties for the selected class.
4. Click the **Edit full synonym** button  in the upper right area of the Full Synonym panel.
5. In the Edit FULL\_SYN Annotation Property window (shown in [Figure 6.18](#)), change information as necessary (for example, *term name* or *term group*).

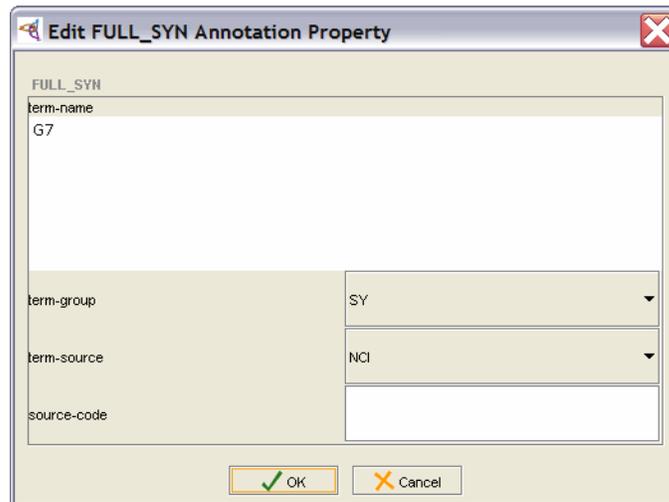


Figure 6.18 Edit FULL\_SYN Annotation Property window

6. Click **OK** to accept the change and close the editing window.
 

**Note:** If you change the *term-group* property to a type that is designated as a preferred term in the NCI editing rules, an error message appears. If this happens, repeat this procedure and select another term group.
7. Click the **Save** button to accept the change, then close any confirmation messages that appear.

## Deleting a Full Synonym

To delete a full synonym, follow these steps:

1. Select the class from which you want to delete a full synonym in the Class Browser on the left.
2. Click the **Edit** tab on the right 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 accept the change, then close any confirmation messages that appear.

## Modifying a Definition

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

1. Select the class to be modified in the Class Browser on the left.
2. Click the **Edit** tab on the right if it is not already displayed.
3. Click the **Basic Data** subtab.
4. Click the **Modify selected annotation** button , located in the upper right area of the Definitions panel.
5. In the Edit DEFINITION Annotation Property window (shown in [Figure 6.19](#)), edit the text in the **def-definition** field.

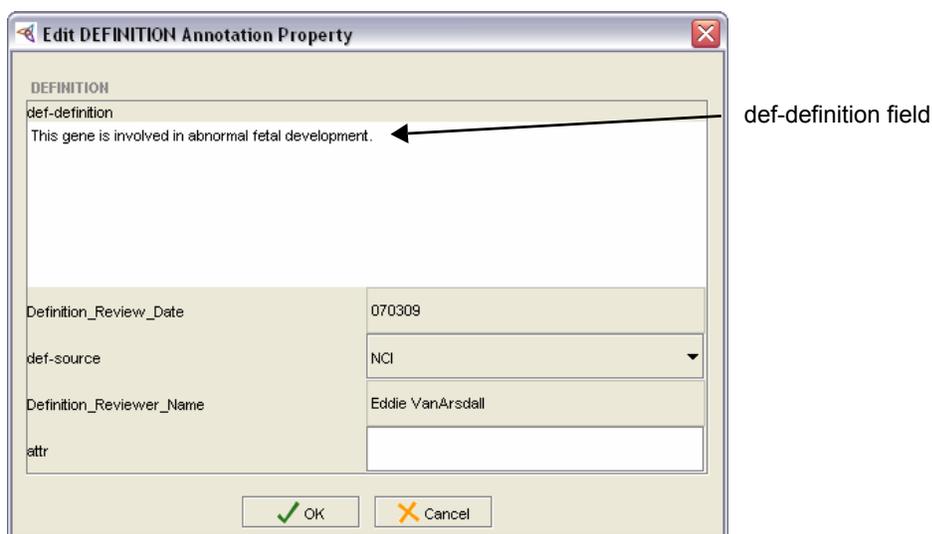


Figure 6.19 Edit DEFINITION Annotation Property window

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

**Note:** A built-in utility checks for extra spaces in the definition, so if you included extra spaces, you will receive the following error: *Unable to save <Concept Name> -- Cannot have non-printable character.* If this happens, repeat this procedure, remove the spaces, and save again.

7. Click the **Save** button to accept the change, then close any confirmation messages that appear.

## Asserting Relations

### In This Section...

- [Adding a Simple Restriction](#) on this page
- [Modifying a Restriction](#) on page 111
- [Deleting a Restriction](#) on page 114
- [Adding a Role Group](#) on page 115

**Note:** For more information about roles and role groups, See [Chapter 2, Description Logic and the NCI Thesaurus Semantic Model](#).

### Adding a Simple Restriction

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

1. Select the class to be modified in the Class Browser on the left.
2. Click the **Edit** tab on the right if it isn't already displayed.
3. Click the **Relations** subtab on the **Edit** tab.
4. Click the **Add a restriction/group** button  in the upper right area of the Restrictions/Groups panel, as shown in [Figure 6.20](#).

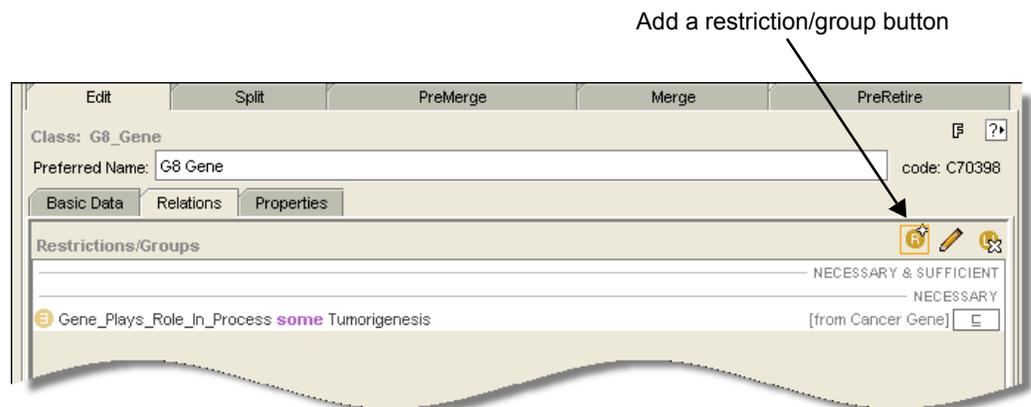


Figure 6.20 Edit tab - Relations subtab - Restrictions panel

5. In the Create a Restriction window (shown in [Figure 6.21](#) on page 108), click the **Create a Role** button , the first of the three buttons in the right side of the window.

The Create a Role button is identical to the button used in the last step.

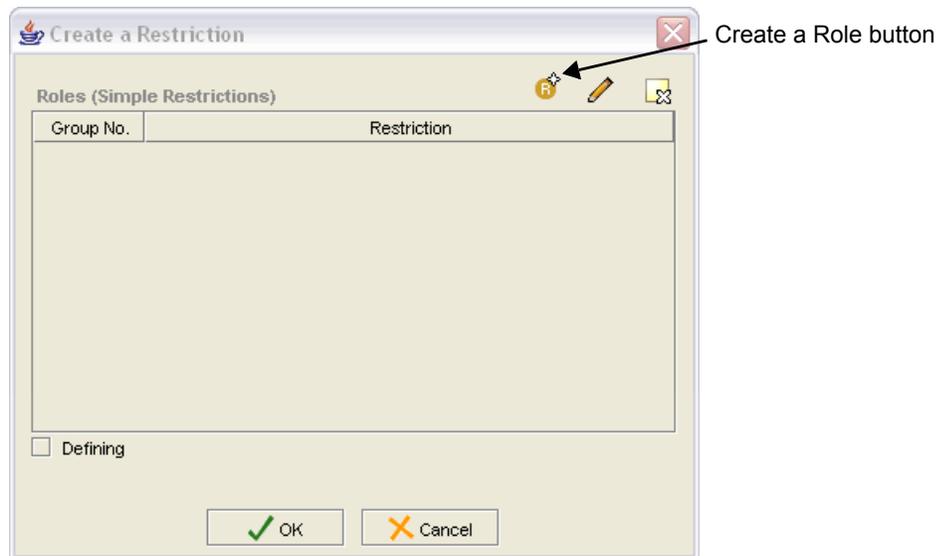


Figure 6.21 Create a Restriction window (first of two windows)

Clicking the Create a Role button opens a second window in front of the first window. Note that both of the open windows are titled *Create a Restriction*.

6. In the second Create a Restriction window (the one shown in front), follow these steps:
  - a. Select a property in the **Restricted Property** list.
  - b. Select a modifier in the **Restriction** list (for example, **someValuesFrom**).
  - c. Click the **Select a named class (filler)** button  in the lower right area of the window, as shown in *Figure 6.22*.

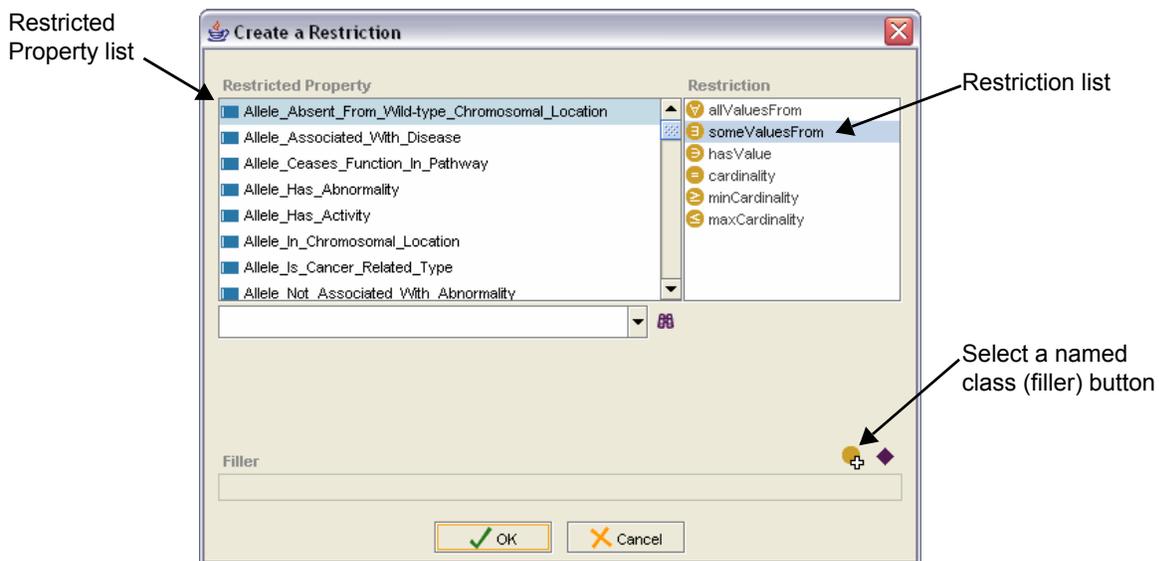


Figure 6.22 Create a Restriction window (second of two windows)

- d. In the Select a named class window (shown in [Figure 6.23](#)),
- select a class from the list, or
  - type a value in the **Search** field and click the **Search** button.

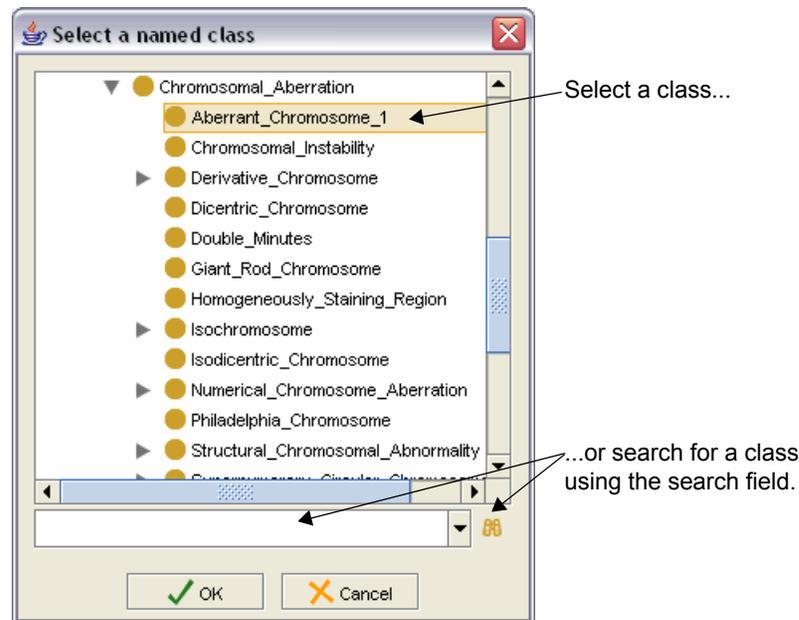


Figure 6.23 Select a named class window

- e. (Search option only) When 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.
- f. Click **OK** to close the Select a named class window.

**Note:** Remember, both of the Create a Restriction windows are still open. In the second of the two windows (the one currently in front), the **Filler** field now shows the class that you selected in Step *d*. (See [Figure 6.24](#).)

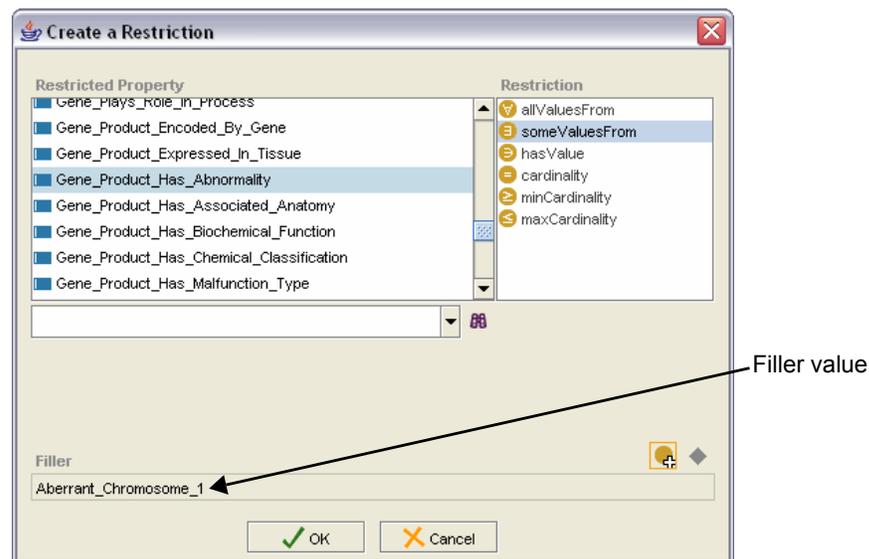


Figure 6.24 Create a Restriction window with Filler value

- g. Click **OK** to close the second Create a Restriction window. The original window now shows the new restriction.
7. (Optional) If the current role is a defining role, check the **Defining** box in the lower left of the window, as shown in [Figure 6.25](#).

**Note:** To add a role group or class expression, see [Adding a Role Group](#) on page 115.

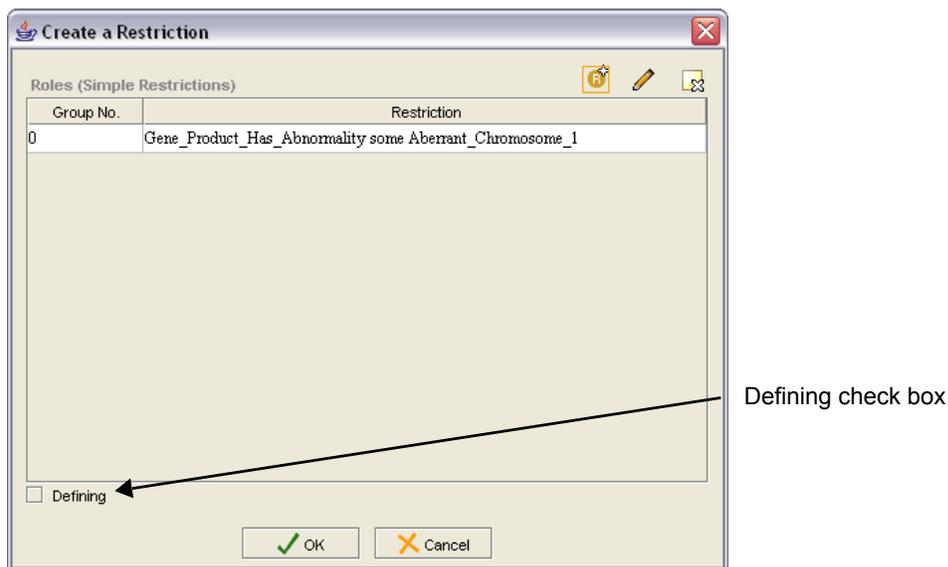


Figure 6.25 Newly added restriction

8. Click **OK**. The new restriction now appears in the Restrictions/Groups list.
9. Click the **Save** button to accept the new restriction, then close any confirmation messages that appear.

## Modifying a Restriction

To modify a restriction, follow these steps:

1. Select the class to be modified in the Class Browser on the left.
2. Click the **Edit** tab on the right if it isn't already displayed.
3. Click the **Relations** subtab.
4. Select the restriction to be edited on the Restrictions/Groups panel.
5. Click the **Edit a restriction/group** button  in the upper right area of the panel.

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

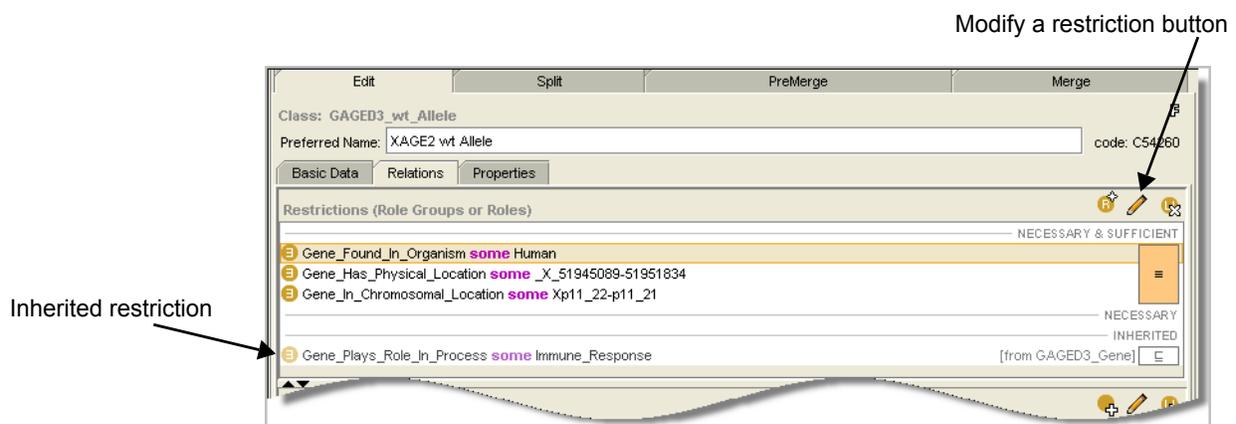


Figure 6.26 Inherited restriction

6. In the Edit a Restriction window, select the restriction to be modified.
7. Click the **Modify a role (simple restriction)** button  in the upper right area of the window. (See [Figure 6.27.](#))

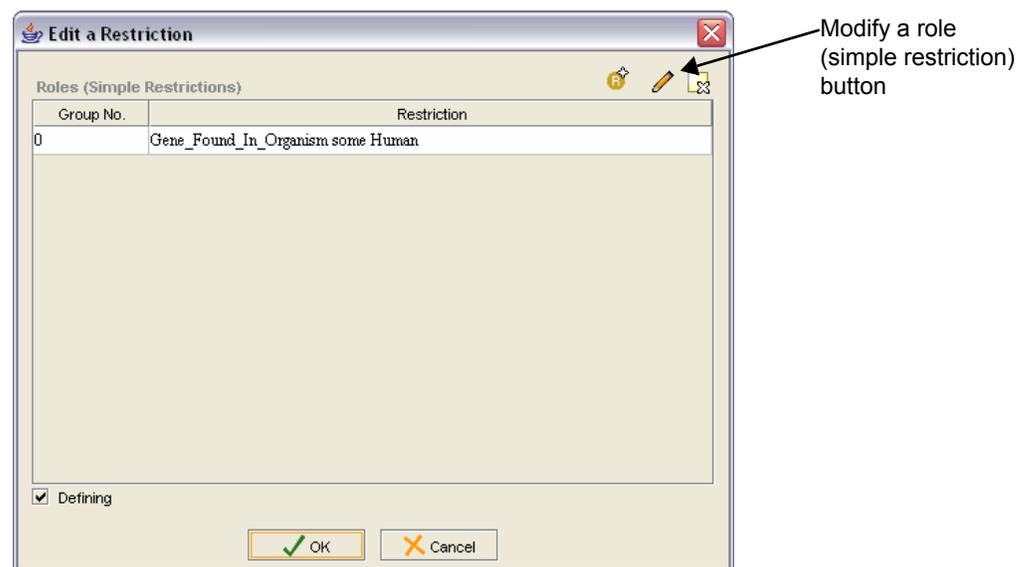


Figure 6.27 Edit a Restriction window

8. In the Modify a Restriction window, do the following:
  - a. Select an item in the **Restricted Property** list.
  - b. Select a modifier in the **Restriction** list.
  - c. Click the **Select a named class (filler)** button  in the lower right area of the window, as shown in *Figure 6.28*.

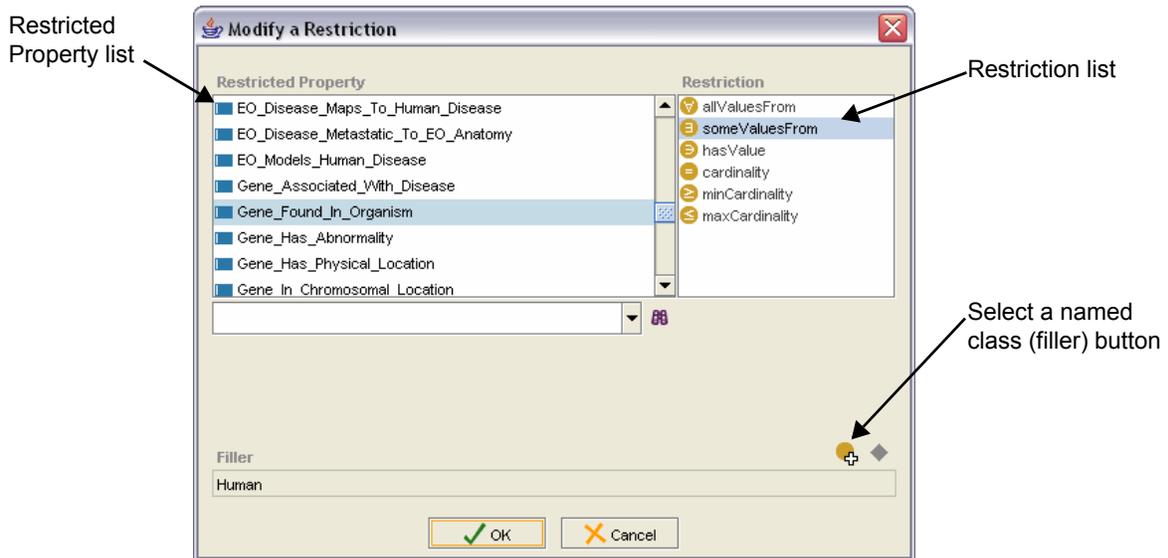


Figure 6.28 Modify a Restriction window

- d. In the Select a named class window, (shown in *Figure 6.29*),
  - select a class from the list, or
  - type a value in the **Search** field and click the **Search** button.

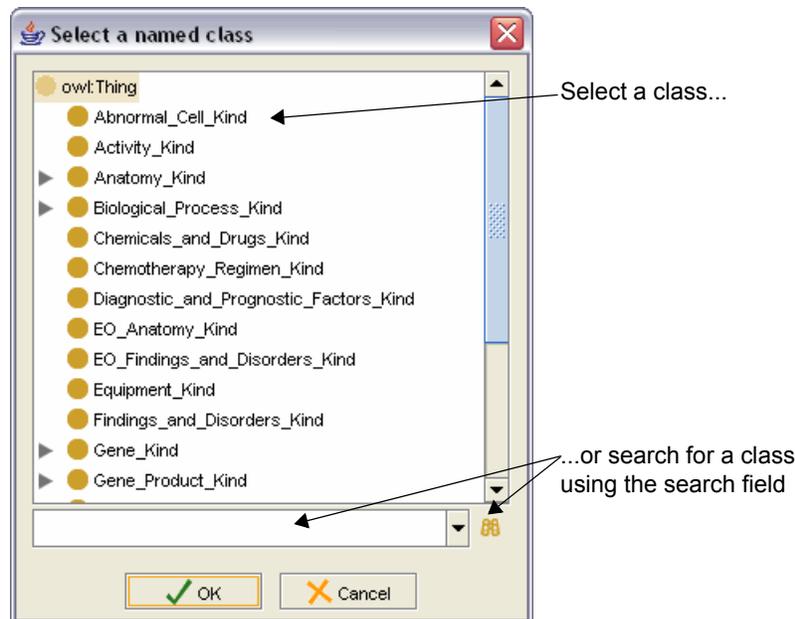


Figure 6.29 Select a named class window

- e. (Search option only) When 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.
- f. Click **OK** to close the Select a named class window. The selected value now appears in the **Filler** field.

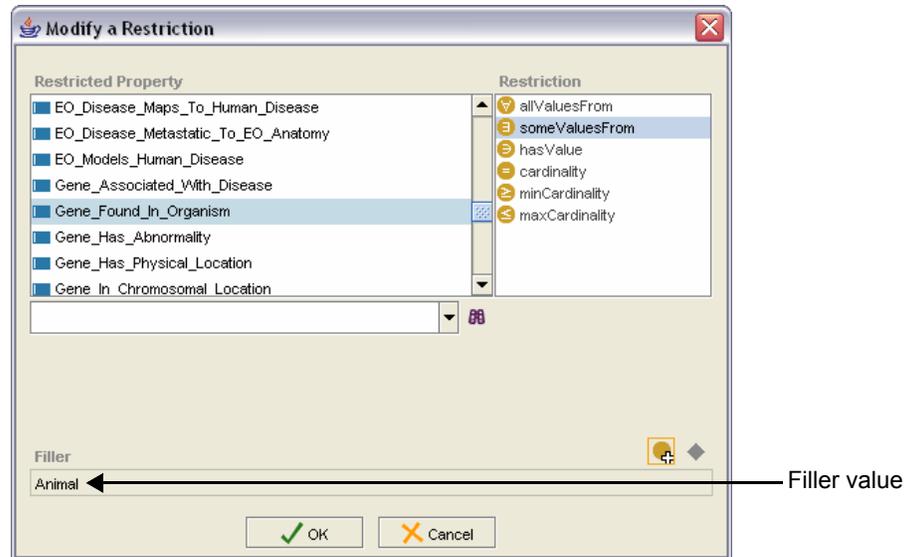


Figure 6.30 Modify a Restriction window with new Filler value

9. Click **OK** to close the Modify a Restriction window. The new restriction now appears in the original window.

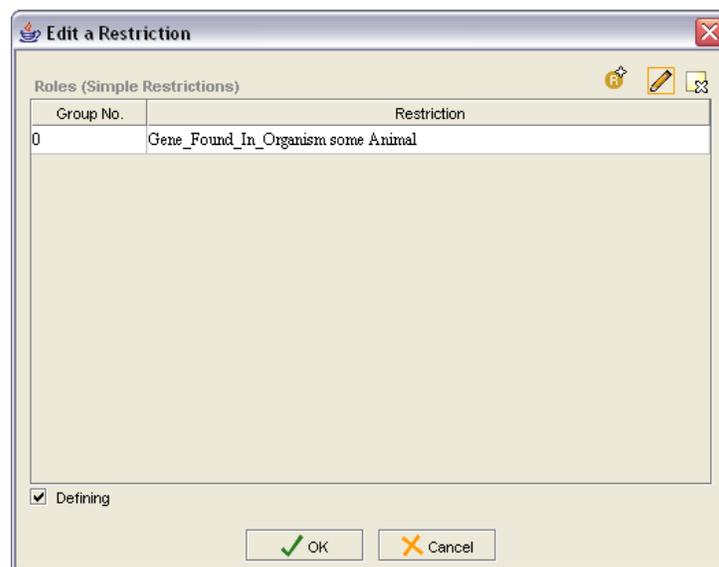


Figure 6.31 Newly modified restriction

10. Click **OK** to close the Edit a Restriction window.
11. Click the **Save** button to accept the change, then close any confirmation messages that appear.

## Deleting a Restriction

To delete a restriction, follow these steps:

1. Select the class from which you want to delete a restriction in the Class Browser on the left.
2. Click the **Edit** tab on the right 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 , located 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.

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

7. Click the **Save** button to accept the change, then close any confirmation messages that appear.

## Adding a Role Group

To add a role group, follow the steps in this section.

### Adding Multiple Restrictions

1. Select the class to which you want to add a role group in the Class Browser on the left.
2. Click the **Edit** tab on the right 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, as shown in [Figure 6.32](#).

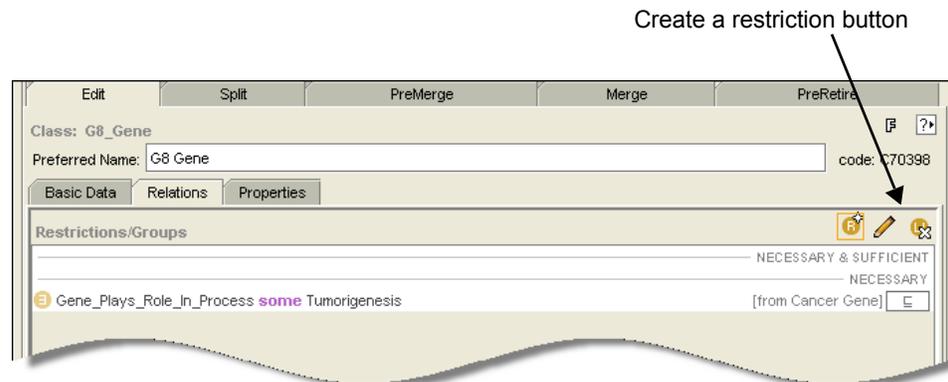


Figure 6.32 Edit tab - Relations subtab - Restrictions panel

5. In the Create a Restriction window (shown in [Figure 6.33](#)), click the **Create a Role** button , the first of the three buttons in the right side of the window. The Create a Role button is identical to the button used in the last step.

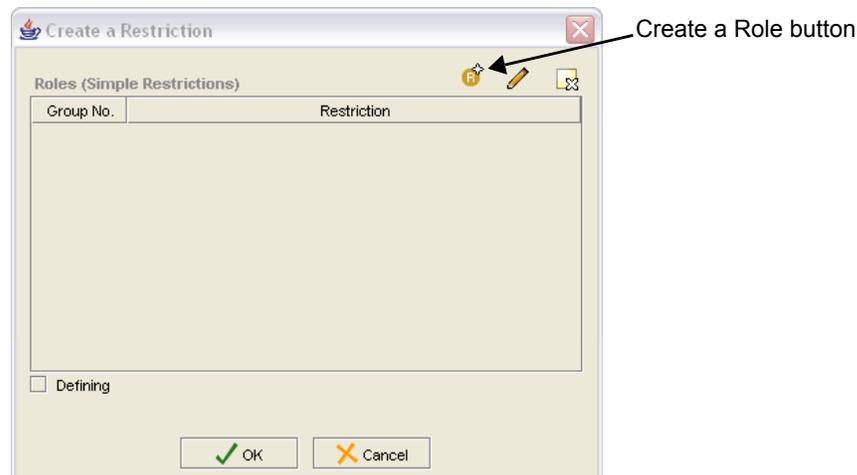


Figure 6.33 Create a Restriction window (first of two windows)

Clicking the **Create a Role** button opens a second window in front of the first window. Note that both of the open windows are titled *Create a Restriction*.

6. In the second Create a Restriction window, follow these steps:
  - a. Select an item in the **Restricted Property** list.
  - b. Select a modifier in the **Restriction** list (for example, **someValuesFrom**).
  - c. Click the **Select a named class (filler)** button  in the lower right area of the window, as shown in [Figure 6.34](#).

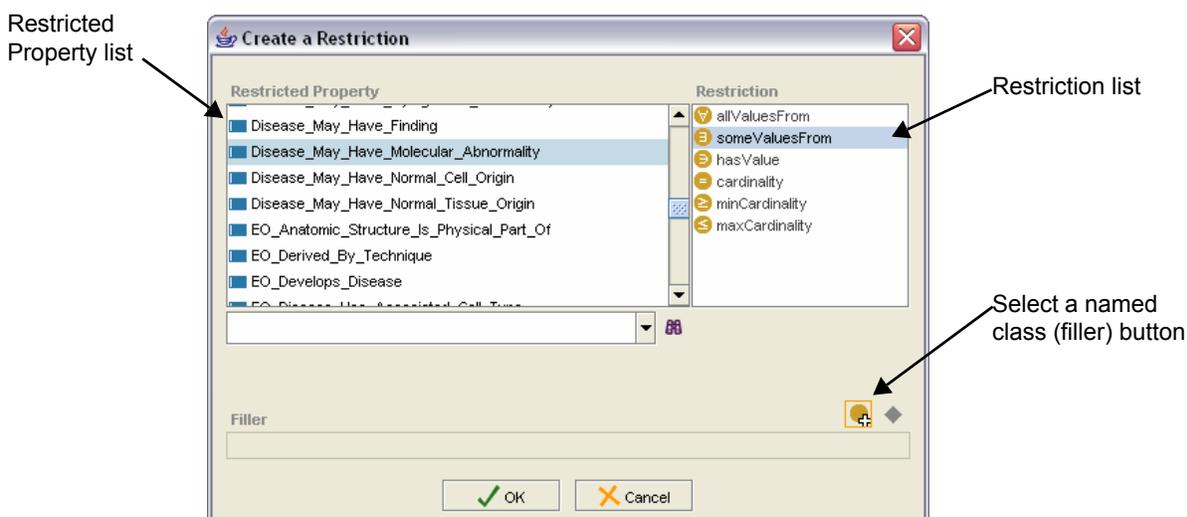


Figure 6.34 Create a Restriction window (second of two windows)

- d. In the Select a named class window (shown in [Figure 6.35](#)),
  - select a class from the list, or
  - type a value in the **Search** field and click the **Search** button.

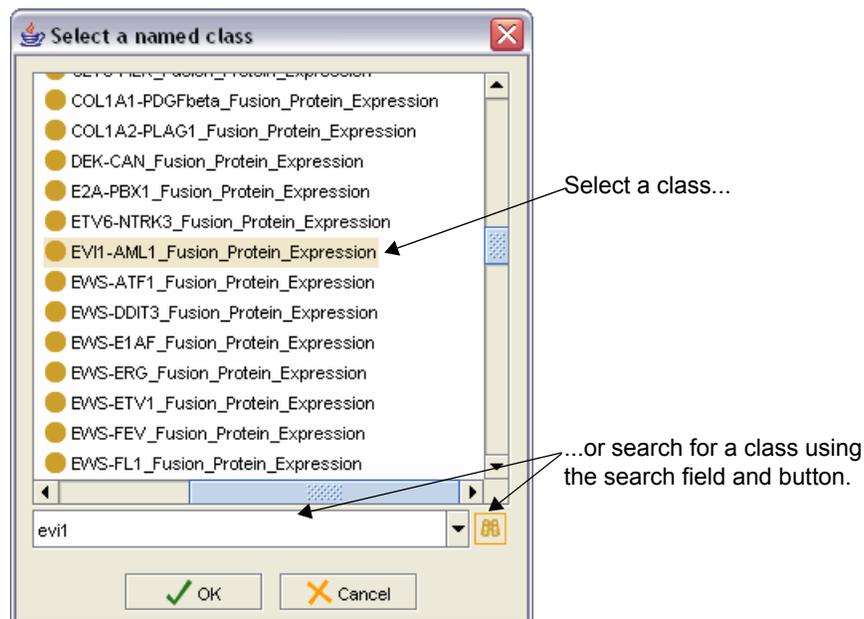
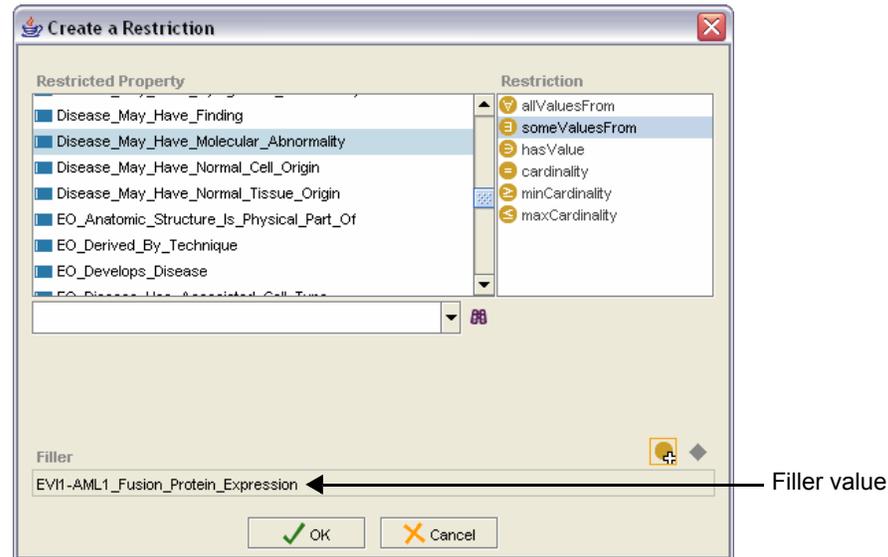


Figure 6.35 Select a named class window

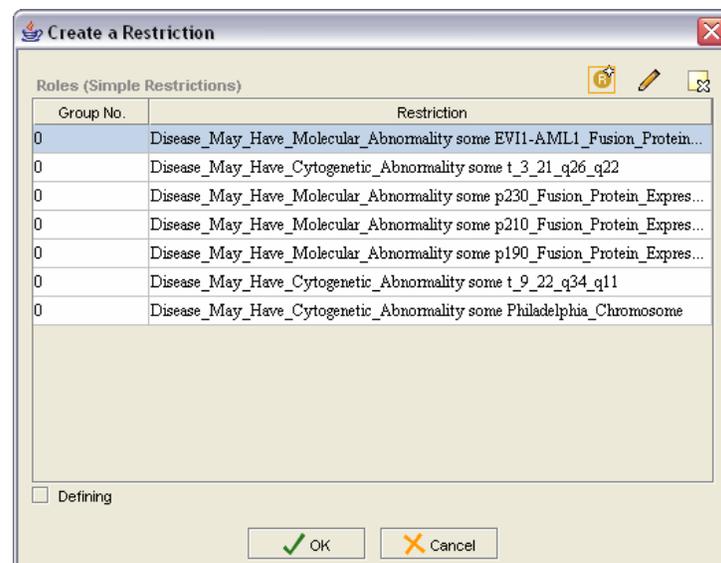
- e. (Search option only) When the Advanced Query window opens, select a search result, then click **OK** to close the window and return to the Select a named class window.
- f. Click **OK** to close the Select a named class window.

**Note:** Remember, both of the Create a Restriction windows are still open. In the second of the two windows (the one currently in front), the **Filler** field now shows the value that you selected in Step *d*. (See *Figure 6.36*.)



*Figure 6.36 Create a Restriction window with Filler value*

- g. Repeat Steps 5. through 6. f. to add more restrictions.
- h. Click **OK** to close the second Create a Restriction window. The original window now shows a group of restrictions.



*Figure 6.37 Multiple restrictions*

- 7. Leave this window open and continue to the next section.

## Building a Role Group Expression

In the last section, you created a group of restrictions. To build a role group expression from the group, follow these steps:

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

1. Starting with the first item in the list, double-click the number on the left, under the heading **Group No.**, as shown in [Figure 6.38](#).
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.

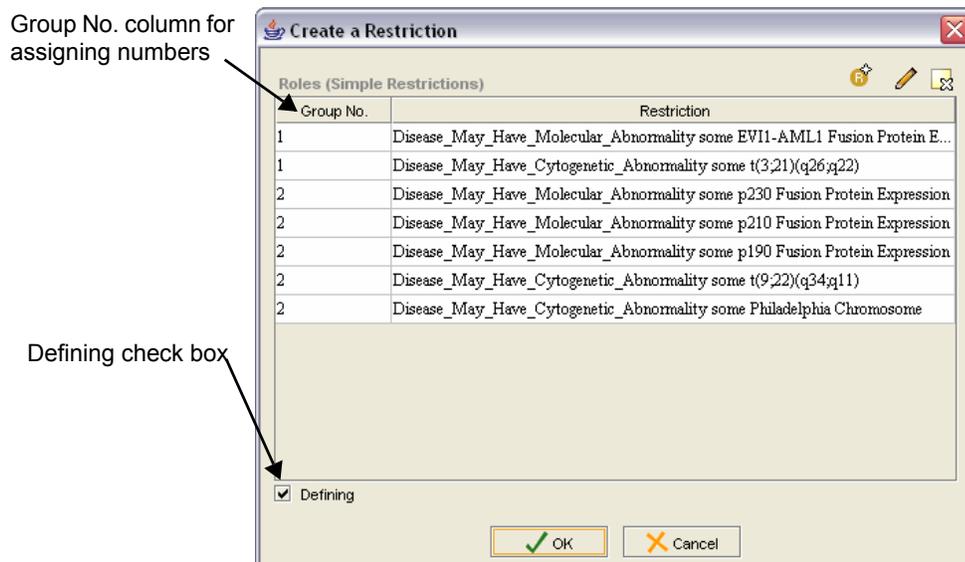


Figure 6.38 Role group example

5. Click **OK**. The new role group now appears in the Relations panel. It is easily identifiable by the length of the expression, as shown in [Figure 6.39](#).

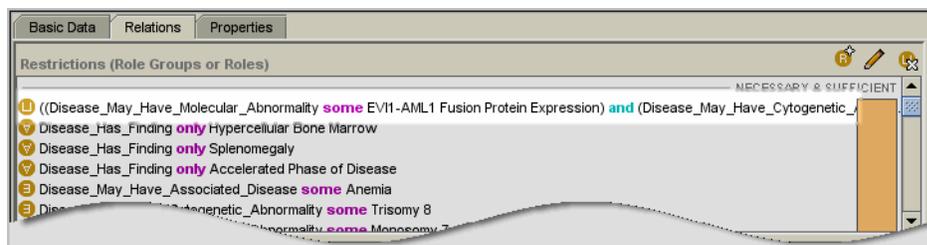


Figure 6.39 Role group expression

**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 at the bottom of the Relations subtab, then close any confirmation messages that appear.

## 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. Follow this path in the Class Browser on the left:

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

2. Click the **Edit** tab on the right if it is not already displayed. The Basic Data tab displays information for the Phagocytosis class.
3. Click the **Relations** subtab.
4. Locate the first of the three buttons  in the upper right area of the Associations panel, as shown in [Figure 6.40](#).

**Tip:** When you hover the mouse pointer over the button, the tool tip reads **Add existing resource as value...**

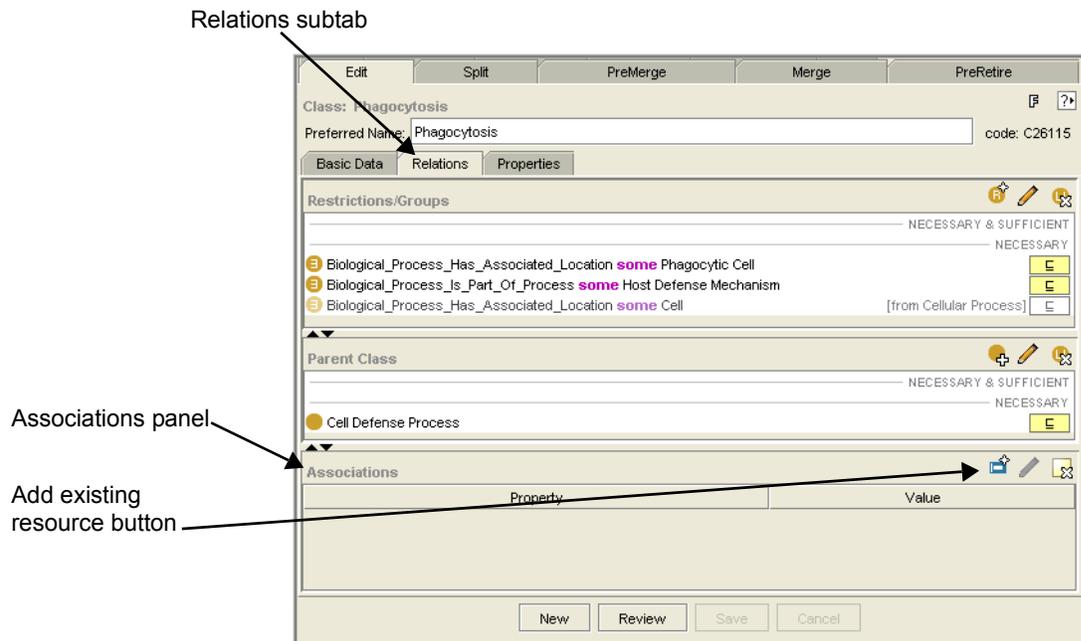
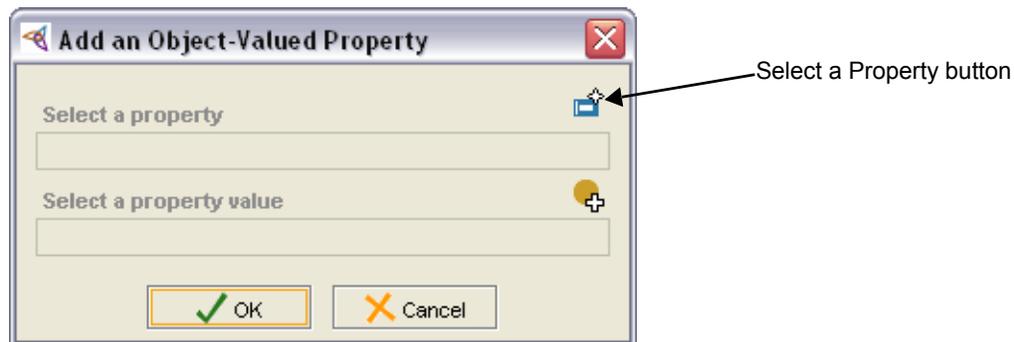


Figure 6.40 Relations subtab for Phagocytosis

5. Click this button to open the Add an Object-Valued Property window.

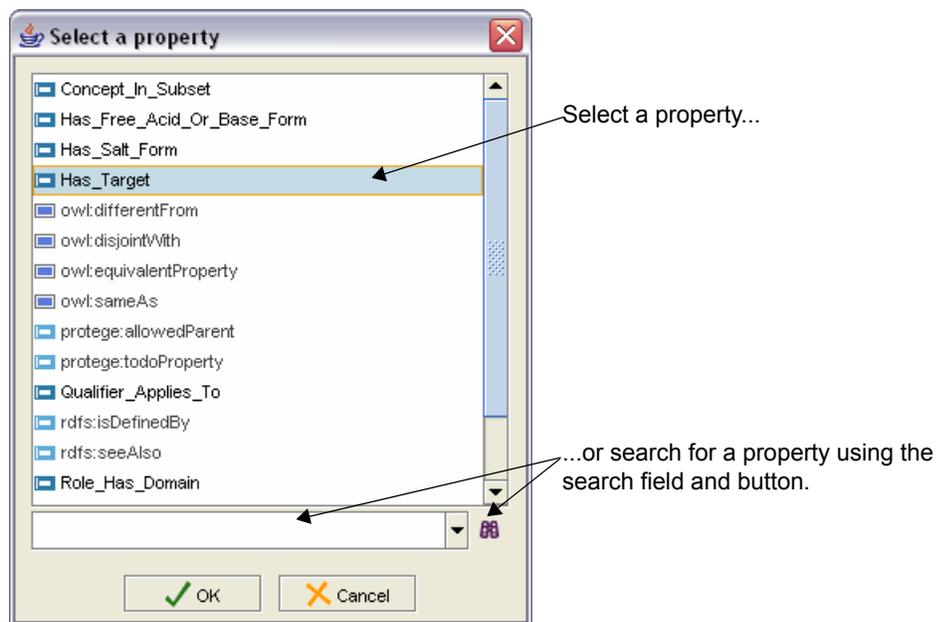
6. In the Add an Object-Valued Property window (shown in *Figure 6.41*), click the **Select a Property** button  in the upper right area.

This button is identical to the button described in Step 4. .



*Figure 6.41 Add an Object-Valued Property window*

7. In the Select a Property window,
  - select **Has\_Target**, or
  - type a value in the **Search** field and click the **Search** button.
8. (Search option only) When the Advanced Query window opens, select a search result, then click **OK** to close the window and return to the Select a property window.



*Figure 6.42 Select a Property window*

9. 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, as shown in [Figure 6.43](#).

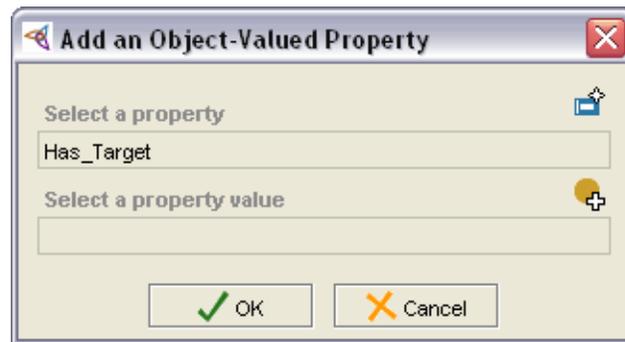


Figure 6.43 Select a property value added

10. Click the **Select a property value** button .
11. In the Select a property value window, follow this path:  
**Anatomy\_Kind > Anatomic\_Structure\_System\_or\_Substance > Microanatomic Structure > Cell**
12. Click **OK**. As shown in [Figure 6.44](#), the Add an Object-Valued Property window now shows the value *Cell* in the **Select a property value** field.



Figure 6.44 Add an Object-Valued Property window with stored values

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

The Associations panel now displays the *Has\_Target* property and the *Cell* property value, as shown in [Figure 6.45](#).

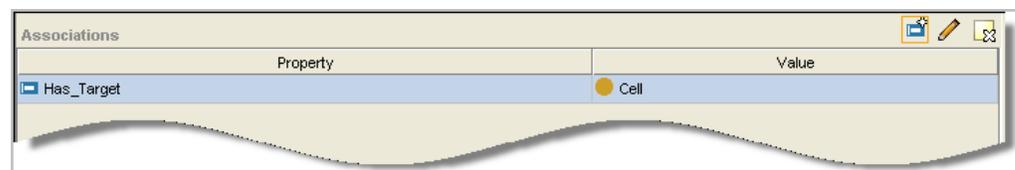


Figure 6.45 Associations panel

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



## CHAPTER 7

# USING ADVANCED EDITING FEATURES

This chapter explains how to perform the more advanced Protégé procedures. The use of the term *advanced* doesn't necessarily mean that the procedures are more difficult to perform; it simply means that they may not be performed by all users.

For example, some of the features detailed in this chapter are used only by workflow managers. The beginning of each procedural section notes whether the procedure is restricted to specific users.

This chapter includes the following topics:

- *Splitting a Class* on page 124
- *Merging Classes* on page 127
- *Retiring a Class* on page 131
- *Using the Report Writer* on page 136
- *Loading a Batch of Classes for Editing* on page 142
- *Editing a Batch of Classes* on page 146
- *Generating a Partonomy Tree* on page 150
- *Copying a Class* on page 152

## Splitting a Class

### When to Consider Splitting a Class

Classes are typically *split* when an editor determines 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.

### Steps for Splitting a Class

To split a class, follow these steps:

1. Select the class to be split in the Class Browser on the left.
2. Click the **Split** tab on the right.
3. Drag the selected class from the Class Browser into the **Existing Concept** pane (the upper pane on the **Split** tab), as shown in [Figure 7.1](#).
4. Click the **Split** button at the bottom of the tab.

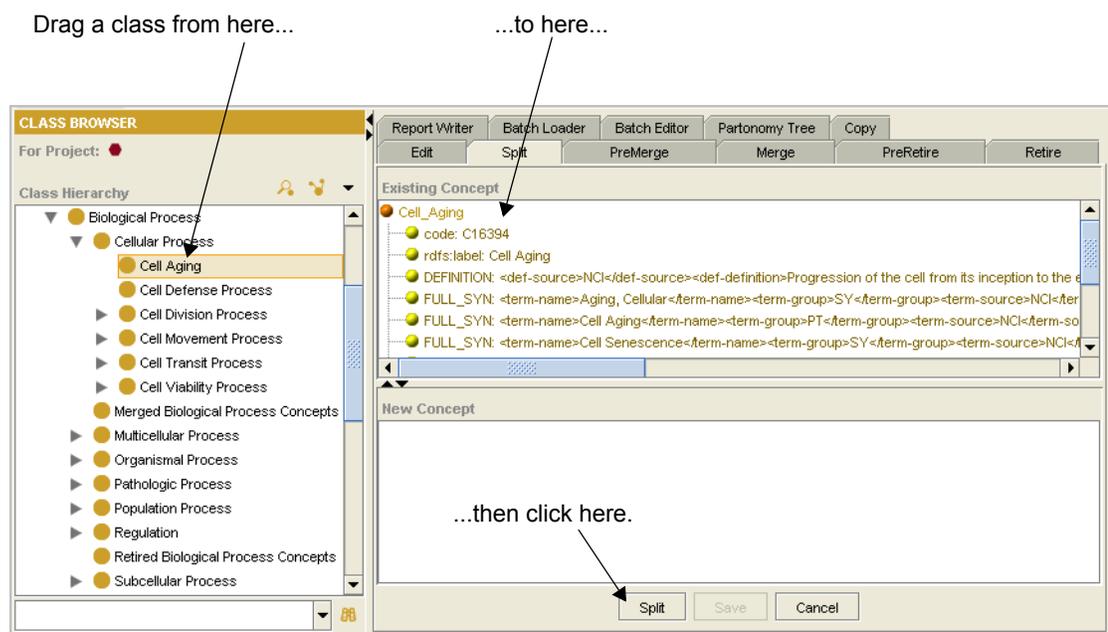


Figure 7.1 Existing Concept pane

- In the Enter Class Identifiers window, enter a label with underscores and a preferred name without underscores, as shown in *Figure 7.2*.

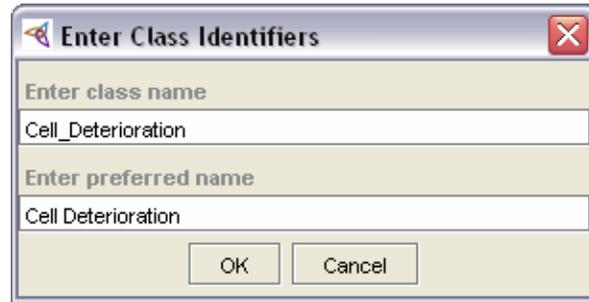


Figure 7.2 Enter Class Identifiers window

- Click **OK**. The new class appears in the lower pane on the **Split** tab, as shown in *Figure 7.3*.

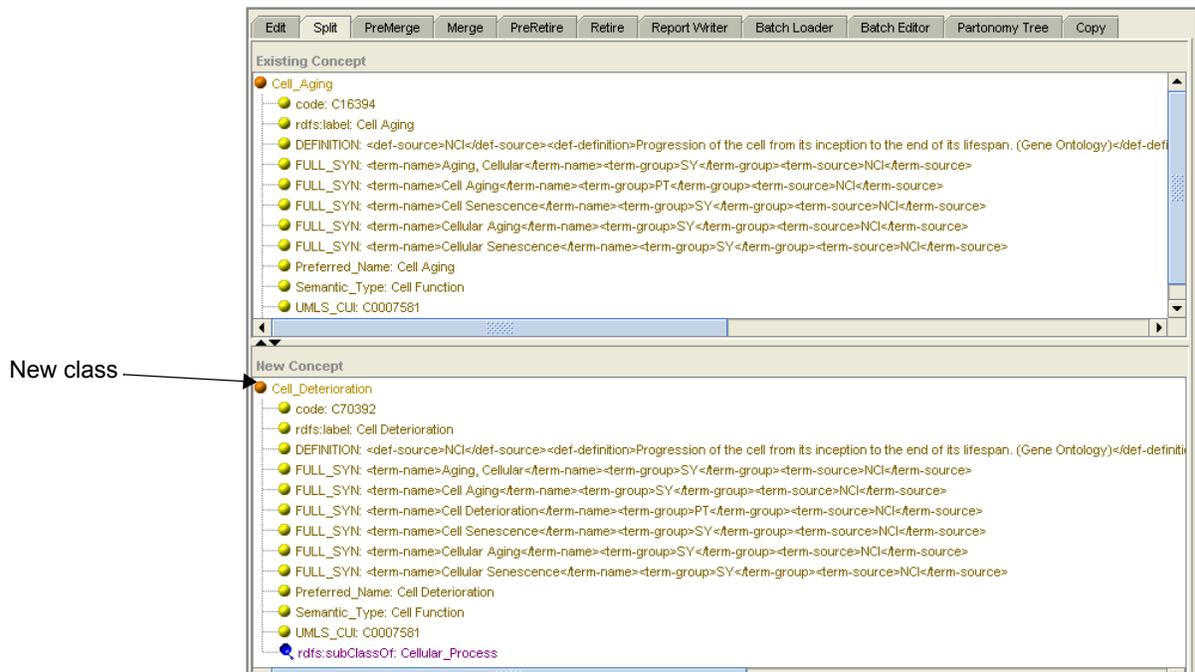


Figure 7.3 Split classes

#### Tips for Editing in the Split Tab Panes

The next steps in this procedure explain how to edit properties for either of the split classes, using a combination of left and right mouse clicks. When editing in the Split tab panes (upper or lower), follow this sequence:

- Select a property or restriction with the **left** mouse button so that the item is highlighted.
- Right-click** on the same selected item, or right-click anywhere in the properties list. Both actions assume that you are editing the property that is highlighted.

**Note:** Right-clicking works only when you click on a selected item or in the property list. It has no effect when you click in the white space surrounding the list.

7. To modify properties for either class, follow these steps:
  - a. Select the property to be modified by clicking it with the **left** mouse button.
  - b. Right-click the same item, then select the appropriate command from the shortcut menu (for example, **Modify Property**). See [Figure 7.4](#).

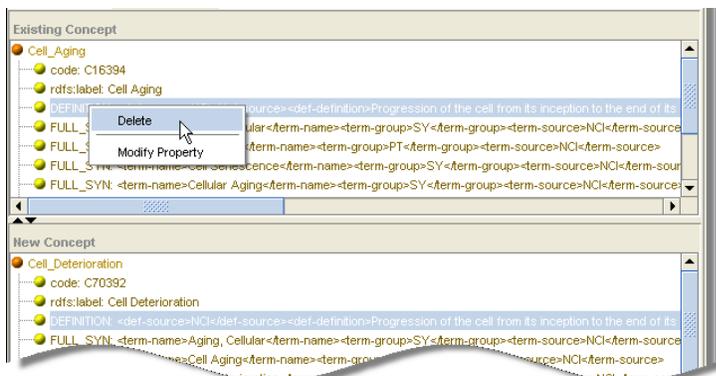


Figure 7.4 Right-click shortcut menu for modifying split classes

- c. In the Edit FULL\_SYN Annotation Property window (or other editing window), edit as needed, then click **OK** to close the window.
8. Click the **Save** button at the bottom of the **Split** tab to accept any changes.
9. Click **OK** to close the confirmation message window. The Existing and New Concept panes are now empty.
10. Locate the new concept in the Class Hierarchy. The new concept appears as a sibling of the existing concept.
11. Click the **Edit** tab to view the properties and restrictions for the new class.
12. On the **Properties** sub-tab, note the *Split\_From* property, which has a value equal to the code of the existing concept.

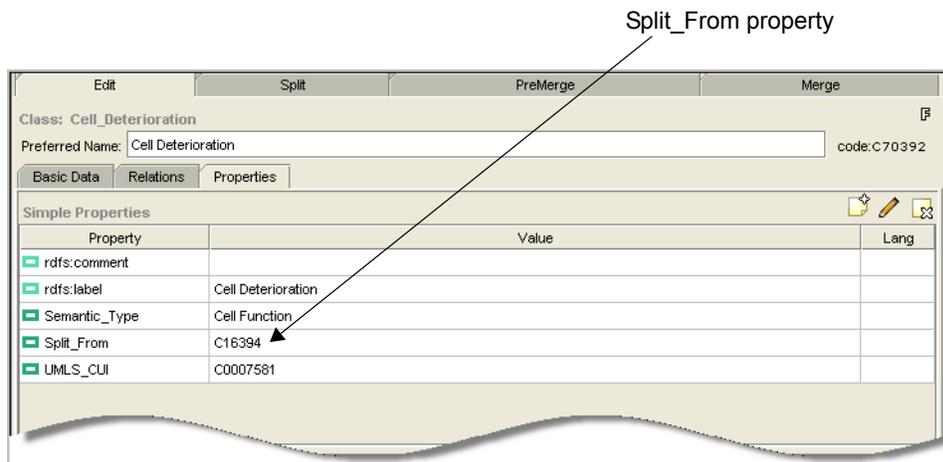


Figure 7.5 Properties of new class

## Merging Classes

---

### When to Consider Merging Classes

Classes are typically *merged* when an editor determines 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 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.

---

### Pre-Merge: Flagging Classes to be Merged

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

1. In the Class Browser on the left, select one of the two classes to be flagged for a pre-merge.
2. Click the **PreMerge** tab.
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** tab.
  - 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** tab.

*Figure 7.6* on page 128 illustrates the paths from the Class Browser to the **Retiring Concept** and **Surviving Concept** panes.

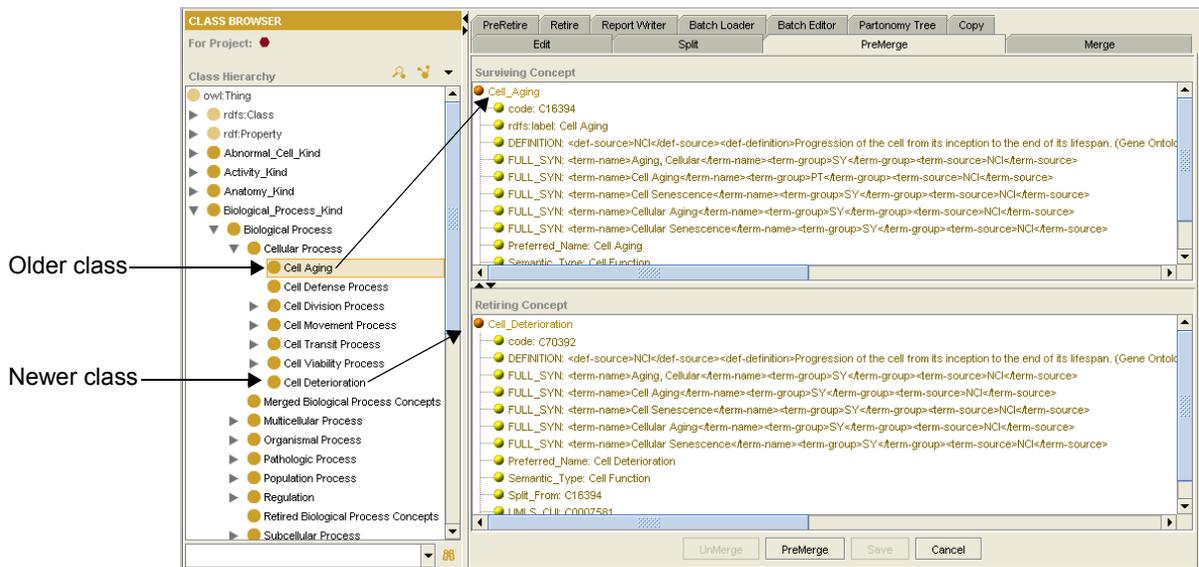


Figure 7.6 PreMerge tab with surviving and retiring concepts

**Tips for Editing in the PreMerge Tab Panes**

The next steps in this procedure explain how to edit properties for either of the PreMerge classes, using a combination of left and right mouse clicks. When editing in the PreMerge tab panes (upper or lower), follow this sequence:

1. Select a property or restriction with the **left** mouse button so that the item is highlighted.
2. **Right-click** on the same selected item, or right-click anywhere in the properties list. Both actions assume that you are editing the property that is highlighted.

**Note:** Right-clicking works only when you click on a selected item or in the property list. It has no effect when you click in the white space surrounding the list.

4. To modify properties for either class, follow these steps:
  - a. Select the property to be modified by clicking it with the **left** mouse button.
  - b. Right-click the same item, then select the appropriate command from the shortcut menu (for example, **Modify Property**). See [Figure 7.7](#).

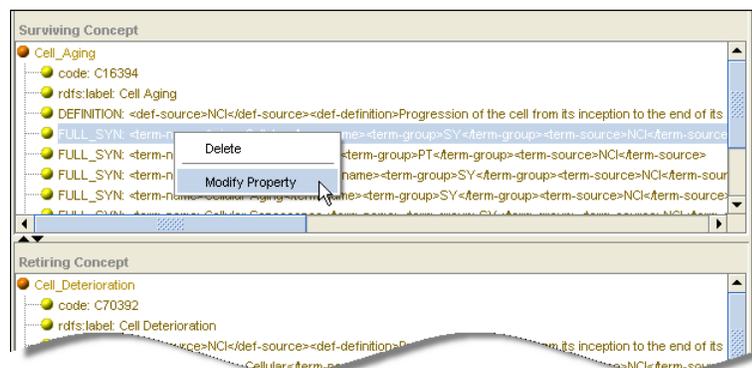


Figure 7.7 Right-click shortcut menu for modifying pre-merged classes

- c. In the Edit FULL\_SYN Annotation Property window (or other editing window), edit as needed, then click **OK** to close the window.
5. Click the **PreMerge** button at the bottom of the **PreMerge** tab.
6. (Required) In the Enter Notes window, read the pre-filled **Editor's Note** and **Design Note**:
  - An editor's note is for internal use only. Since it is not published, you can use it for such things as notes and directions for other editors.
  - A design note provides useful information for all Thesaurus users. For example, it might provide additional information about the meaning and use of a concept.
7. Edit the notes if necessary.

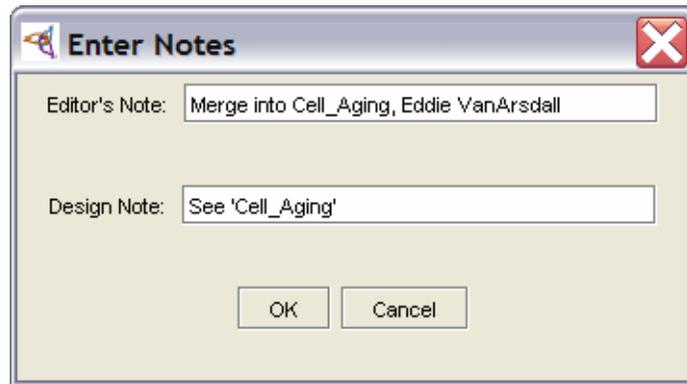


Figure 7.8 Enter Notes window

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

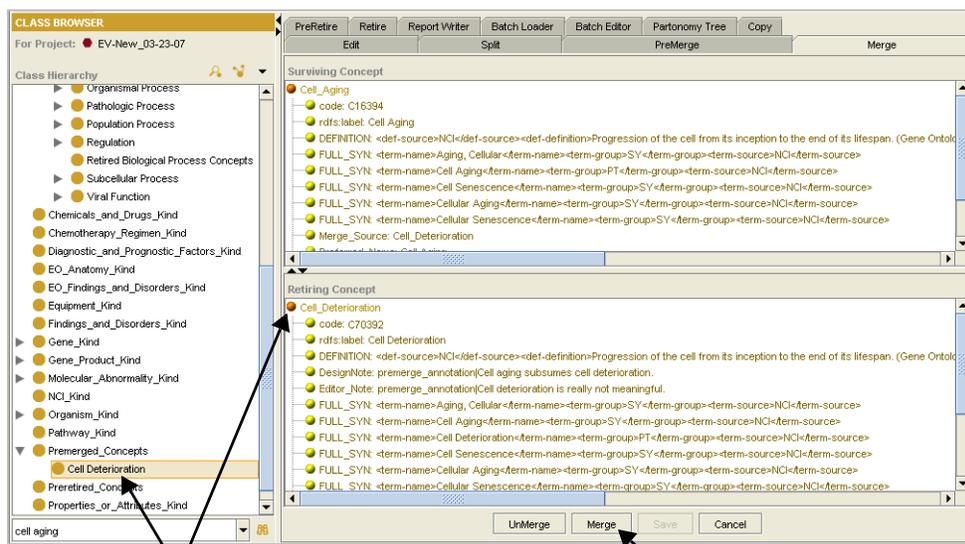
**Note:** The **PreMerge** button is now disabled, and the **UnMerge** button is available. If you want to undo the PreMerge flag, click the **UnMerge** button.
9. Click the **Save** button to accept the change, then close any confirmation messages that appear. The two classes are no longer displayed in the upper and lower PreMerge panes.
 

**Note:** The retiring concept still appears in the Premerged\_Concepts branch of the Class Hierarchy until a workflow manager completes the merge. If you examine the concept properties under **Edit > Properties**, you will see that a *Merge\_Target* property has been added for the concept.

## Merging Flagged Classes (Workflow Managers Only)

To merge flagged concepts, follow these steps:

1. Locate the **Premerged\_Concepts** branch in the Class Browser on the left.
2. Expand the branch.
3. Locate and select the class that will be retired as a result of the merge.
4. Click the **Merge** tab.
5. Drag the selected class on the left into the Retiring Concept (lower) pane on the right. The Surviving Concept and Retiring Concept panes now show their respective classes.
6. Click the **Merge** button at the bottom of the Merge tab.



Drag the selected class into the lower pane...

...then click the Merge button.

Figure 7.9 Merging two classes

**Note:** The **Merge** button is now disabled, and the **UnMerge** button is available. If you want to undo the merge, click the **UnMerge** button.

7. Click the **Save** button to accept the change, then close any confirmation messages that appear.

The two panes on the Merge tab are now empty, and the retired class is no longer visible in the Class Hierarchy.

## Retiring a Class

### 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.

Retiring is a two-step process:

1. An editor applies a pre-retire 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 candidates designated for retirement and either accepts or rejects them.

### Pre-Retire: Flagging a Class for Retirement

#### Selecting a Class

To flag a class for retirement, follow these steps:

1. Select the class to be retired in the Class Browser on the left. This example uses the *Helicobacter* class.
2. Click the **PreRetire** tab on the right.
3. Drag the selected class into the **Retiring Class** panel on the lower left side of the **PreRetire** tab, as shown in [Figure 7.10](#).

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** sub-tab. [Figure 7.10](#) shows the Class Browser and the PreRetire tab with all panes displayed.

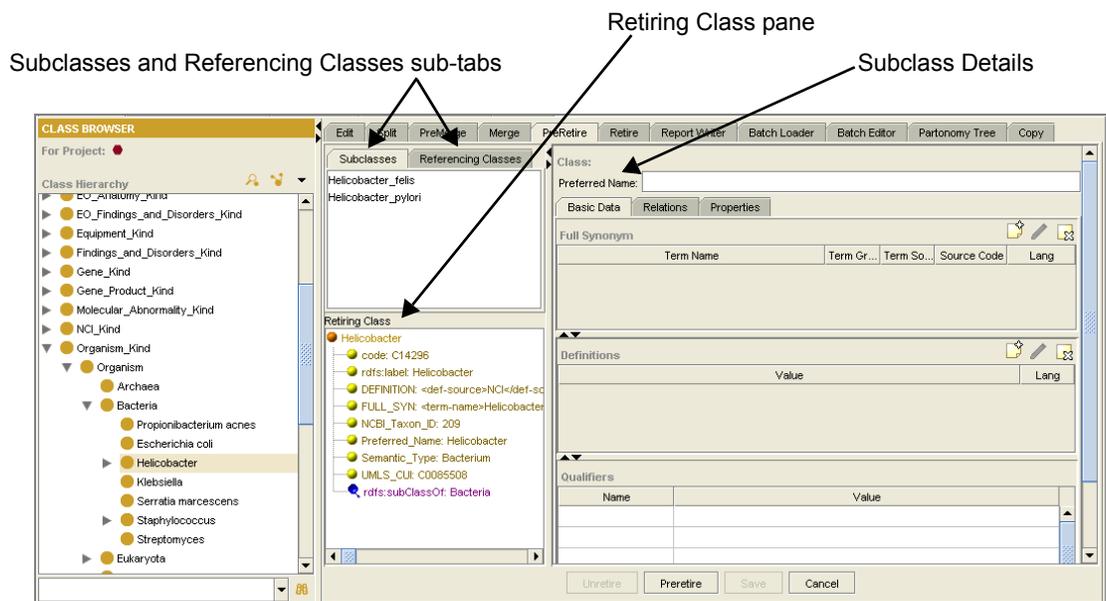


Figure 7.10 PreRetire tab with subclasses

## Re-treeing the Subclasses

When flagging a class for retirement, you first eliminate any dependencies for the class by assigning each of its subclasses to another parent class. To accomplish this, follow these steps for each subclass:

1. Hide the Class Browser by clicking on the **Expand** button on the top right, illustrated in [Figure 7.11](#). This will help you to see all of the details on the middle and right panels.

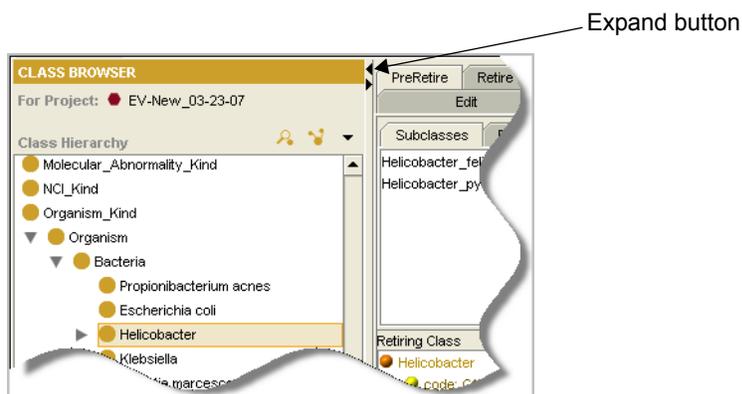


Figure 7.11 Hiding the Class Browser

2. Select a subclass in the Subclasses panel (upper left). As shown in [Figure 7.12](#), detailed information for the selected subclass now appears in the Class Editor pane on the right.

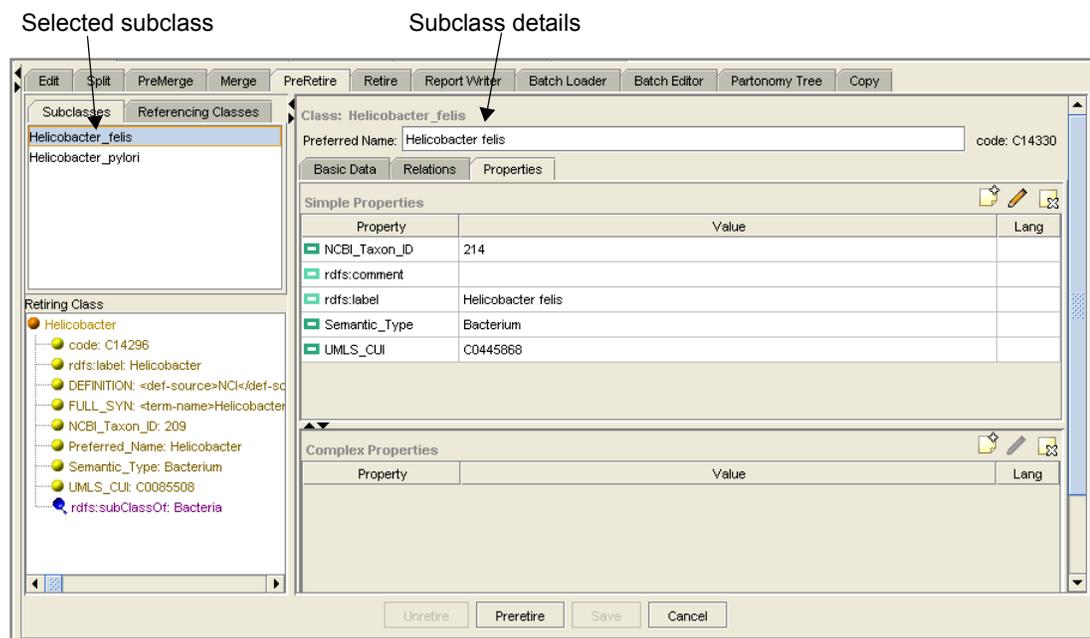
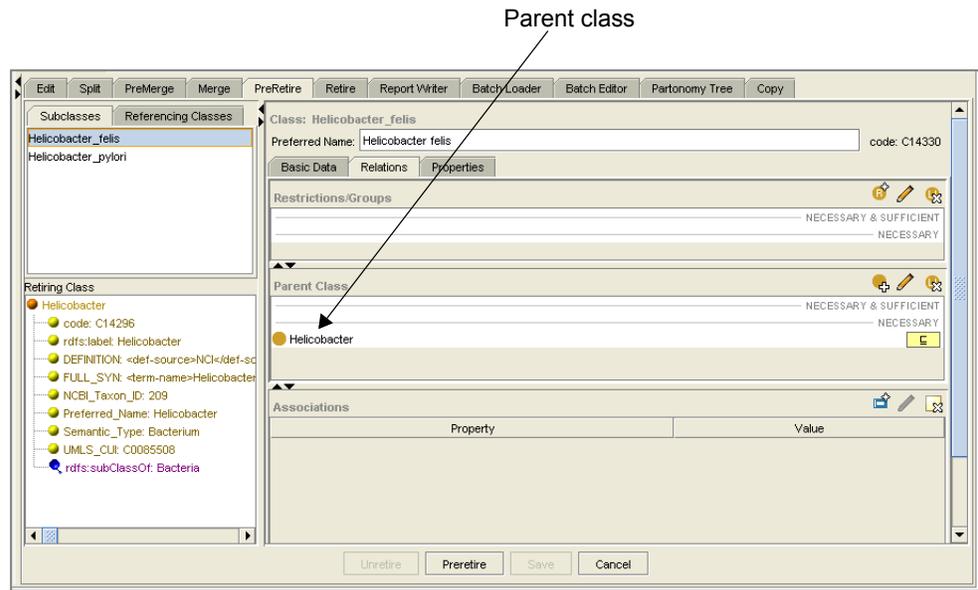


Figure 7.12 Selected subclass with Basic Data subtab displayed

3. To view parent classes for the selected class, click the **Relations** subtab.

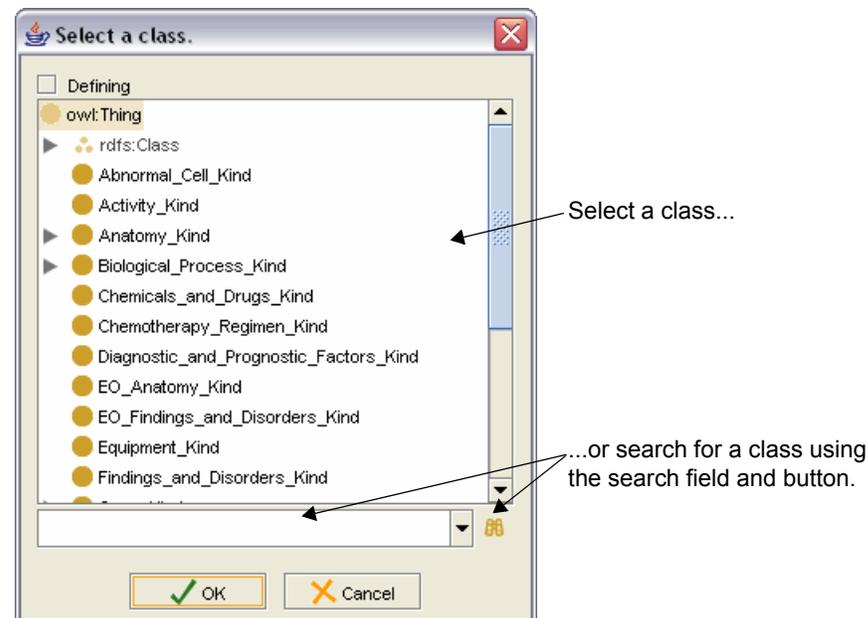
As [Figure 7.13](#) shows, the current example has only one parent class: *Helicobacter*. When only one named parent class exists, you cannot delete it.



*Figure 7.13 Relations tab with one parent class*

To retire 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 panel.
5. In the Select a class window (shown in [Figure 7.14](#)),
  - select a class from the list, or
  - type a value in the **Search** field and click the **Search** button.



*Figure 7.14 Select a class window*

6. (Search option only) When the Advanced Search window opens, select a search result, then click **OK** to close the window and return to the Select a class window.
7. Click **OK** to close the Select a class window. In the current example, there are now two parent classes: *Bacteria* and *Helicobacter*.
8. Select the class to be deleted. For the current example, select the *Helicobacter* class.
9. Click the **Delete selected row** button  in the upper right area of the Parent Class panel.
10. When the Confirm Delete message appears, click **Yes** to accept the deletion.

**Note:** The Basic Data, Relations, and Properties subtabs are nested inside of the PreRetire tab, so you may not notice that there are two separate sets of buttons at the bottom, as shown in [Figure 7.15](#). In the next step, make sure that you click the **Save** button at the bottom of the *inner* tab 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* tab. You will then be asked to save the change using the *outer* **Save** button.

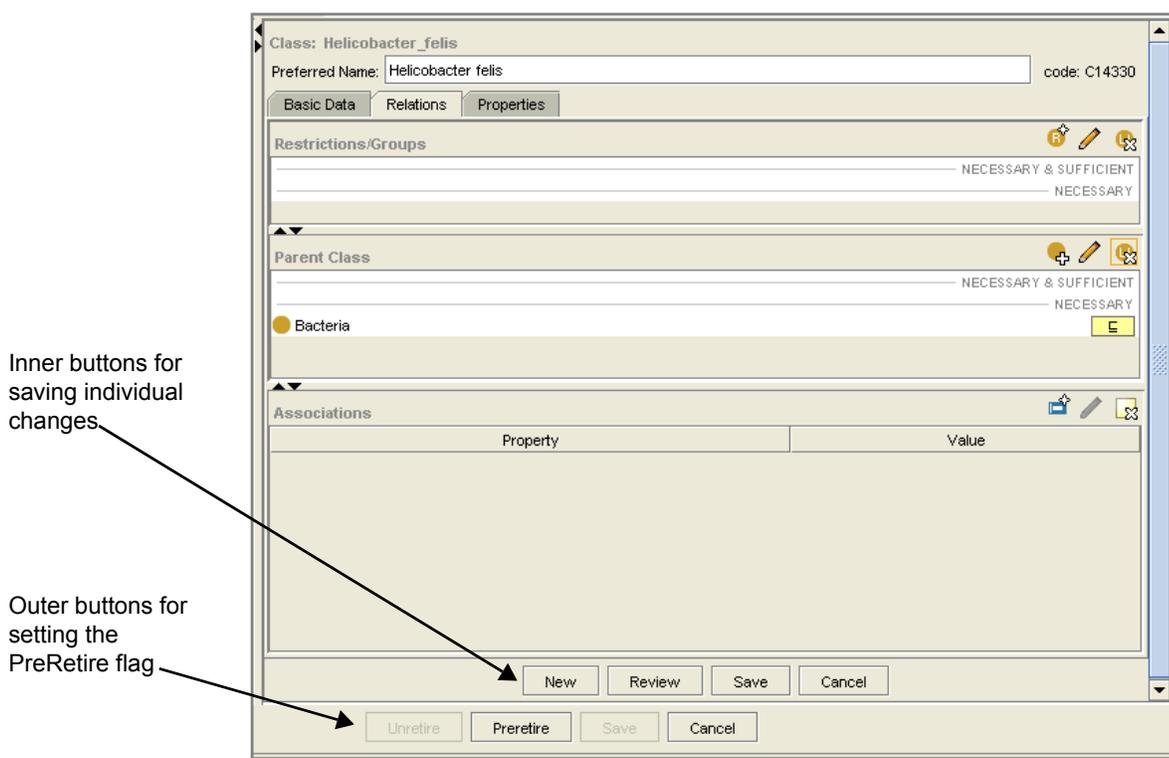


Figure 7.15 Buttons sets for inner, nested subtabs and outer tab

11. Click the *inner* **Save** button at the bottom of the Relations subtab to accept the change. The selected subclass disappears from the Subclasses list.
12. Click **OK** to close the confirmation message window.
13. Repeat steps 2. through 12. for all classes shown in the Subclasses panel. When you are finished, there should be no subclasses shown.

## Completing the PreRetire Task

To pre-retire the concept, follow these steps:

1. Click the **PreRetire** button at the bottom of the **PreRetire** tab.
2. (Required) In the Enter Notes window, add an **Editor's Note** and a **Design Note**:
  - An editor's note is for internal use only. Since it is not published, you can use it for such things as notes and directions for other editors.
  - A design note provides useful information for all Thesaurus users. For example, it might provide additional information about the meaning and use of a concept.

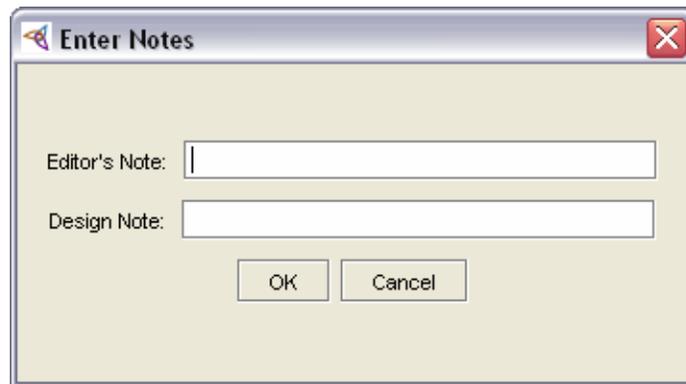


Figure 7.16 Enter Notes window

3. Click **OK** to close the Enter Notes window.
4. Click the *outer* **Save** button—the one at the bottom of the PreRetire tab—to accept the change, then close any confirmation message windows that appear.

The flagged class now appears in the Preretired Concepts branch of the Class Hierarchy.

## Retiring a Flagged Class (Workflow Managers Only)

To retire a concept, follow these steps:

1. Locate the Pretired\_Concepts branch in the Class Browser on the left.
2. Expand the branch, then select the class to be retired.
3. Click the **Retire** tab.
4. Drag the selected class onto the **Retire** tab on the right.
5. Click the **Retire** button at the bottom of the **Retire** tab.
6. If a confirmation message appears, click **OK** to close the message window.
7. Click the **Save** button to accept the change, then close any confirmation messages that appear.

## Using the Report Writer

---

The Report Writer 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. The report output is a text (.txt) file.

### (Optional) Creating an Output Directory and Files

When following the procedure for generating a report, you are required to specify an output file. You can specify a new file while the procedure is running, or you can specify a file that you have already created.

---

**Caution:** Before running reports, create a local or network directory for storing report files. If you do not specify a directory, Protégé stores report files in its own application directory in the *C:\Program Files* folder. Storing report files in the same directory as application files is not a recommended practice. If you decide to delete a report, you risk inadvertently deleting application files that could cause Protégé not to run properly, or not to run at all.

---

While setting up a reports directory, you can also create files for specific class reports. 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. Select a class in the Class Browser on the left. This example uses *Exocrine Gland Fluid or Secretion*.

**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** tab on the right. As shown in [Figure 7.17](#), the Report Writer tab 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.

- With the class still selected in the Class Browser on the left, click the **Generate** button (shown in [Figure 7.17](#)).

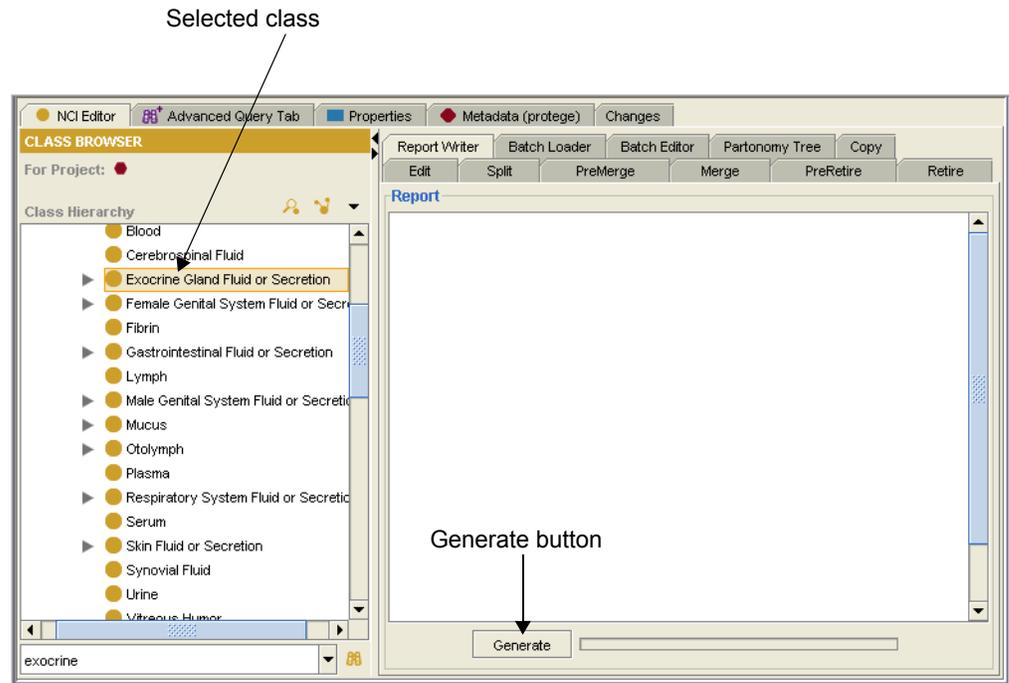


Figure 7.17 Report Writer tab - Generate button

- In the Report Writer window (shown in [Figure 7.18](#)), click the **Browse for File** button in the upper right  area of the window.

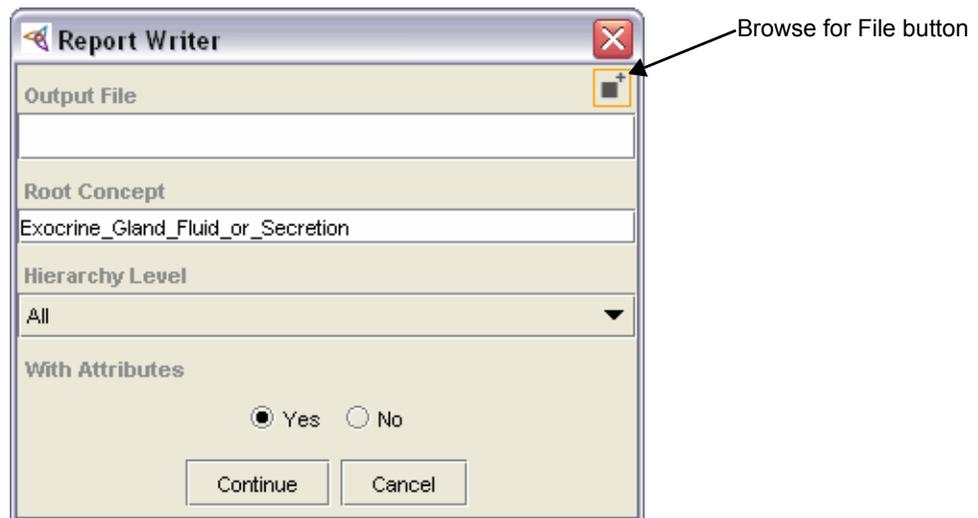
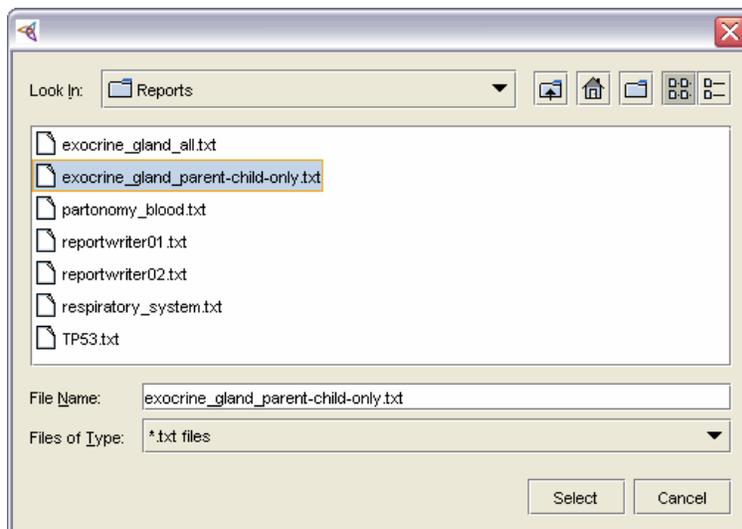


Figure 7.18 Report Writer window for specifying an output file

5. In the File Browser window (shown in *Figure 7.19*), 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.
  - If you are using an existing output file, browse to the appropriate directory, then select the file.



*Figure 7.19 File browser window*

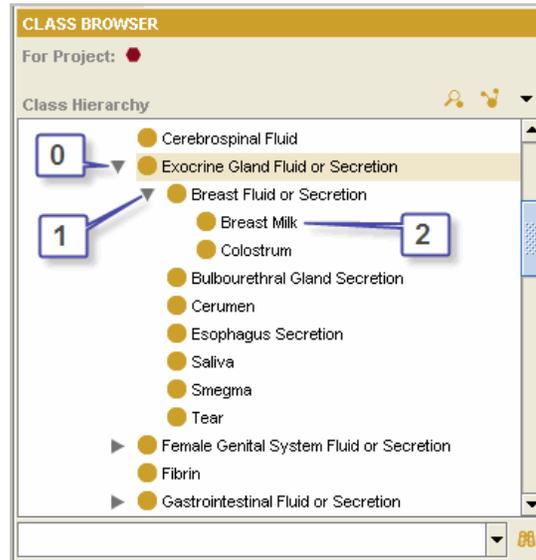
6. Click the **Select** button to close the file browser window. Back in the Report Writer window, the file name now appears in the **Output File** field.
7. Leave the **Root Concept** field as is; do not change the value.
 

**Note:** Although you can type a new root concept name in this field, 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.
8. 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.

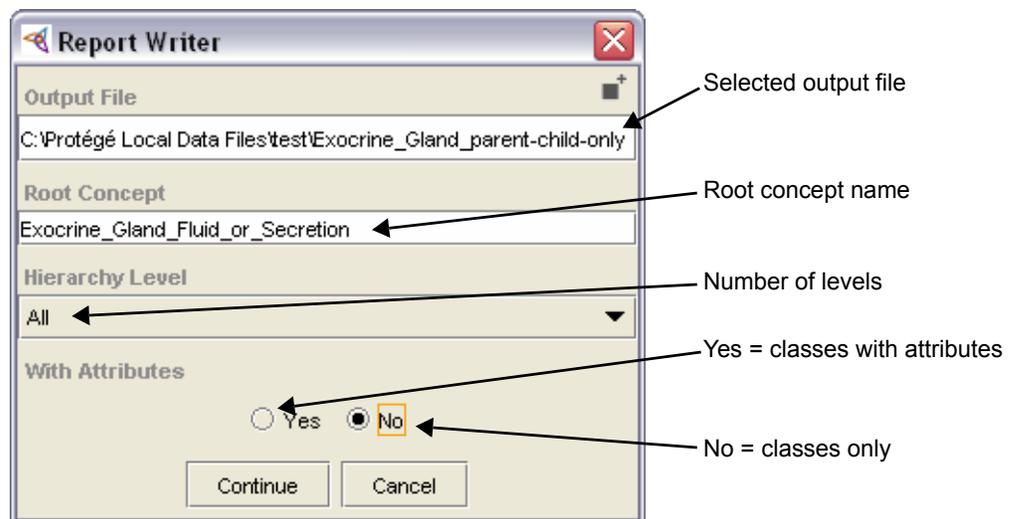
The results of selecting the remaining levels vary according to the number of children that the selected parent has. *Figure 7.20* shows the level numbers of the current example, *Exocrine Gland Fluid or Secretion*.



*Figure 7.20* Level numbers for *Exocrine Gland Fluid or Secretion*

9. 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.

To view the simpler report first, select **No**.



*Figure 7.21* Report Writer window with completed values

10. Click the **Continue** button. The Report Writer status window appears in the upper left corner of your computer screen.

Figure 7.22 shows the Report Writer status window in its initial state.

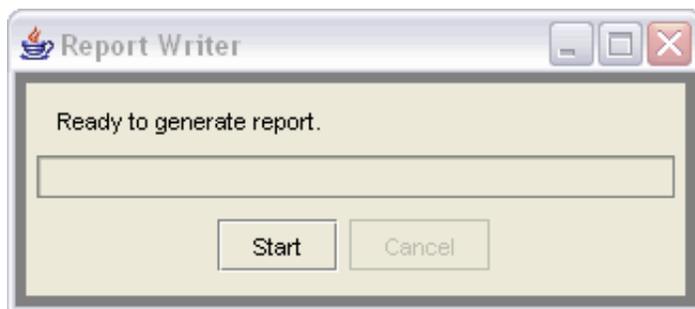


Figure 7.22 Report Writer status window as it initially appears

11. Click the **Start** button. The window displays a progress indicator bar.
12. 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.
13. Using Windows Explorer, browse to the output file that you used for the report.
14. Open the file in a text editor such as Notepad. The file will resemble the sample report shown in [Figure 7.23](#).

This report was created by selecting **No** in the With Attributes section of the Report Writer window. Notice that it shows only classes and subclasses and includes no attributes.

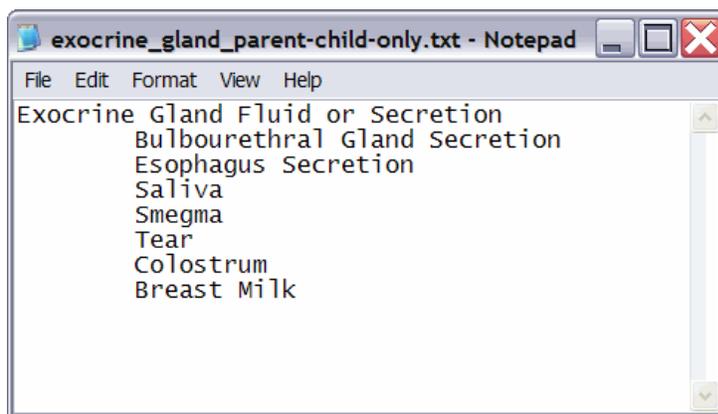


Figure 7.23 Report with classes only - no attributes

The sample report shown in *Figure 7.24* was created by selecting the **Yes** option. This report shows classes, subclasses, properties, and restrictions..

```

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

FULL_SYN: <term-name>Exocrine Gland Fluid or Secretion</term-name><term-group>PT</term-group><term-source>NCI</term-source>

Named Superclass: Body_Fluid_or_Substance

Restriction: Anatomic_Structure_Has_Location some Exocrine System

Bulbourethral Gland Secretion
code: C52554
Preferred_Name: Bulbourethral Gland Secretion

Preferred_Name: Bulbourethral Gland Secretion
Semantic_Type: Body Substance
code: C52554
comment:
rdfs:label: Bulbourethral Gland Secretion

FULL_SYN: <term-name>Bulbourethral Gland Secretion</term-name><term-group>PT</term-group><term-source>NCI</term-source>

```

*Figure 7.24 Report with classes, subclasses, properties, and restrictions*

15. Try generating several reports with and without attributes.

## Loading a Batch of Classes for Editing

The Batch Loader tab enables you to load a batch of classes into Protégé. This is useful when you want to import a large number of classes and then edit them in the application.

You can set up a batch load file in a text editor such as Notepad, or you can set it up in a spreadsheet program such as Excel. In either case, save the file as a tab-delimited text file (.txt).

**Note:** For more information about working with .txt files in Excel, see the Excel online help.

The batch load file requires the following three fields and case-sensitive values:

- **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.

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 7.25](#). As described above, the file should contain three fields: the class name, the preferred name, and the name of the parent class.

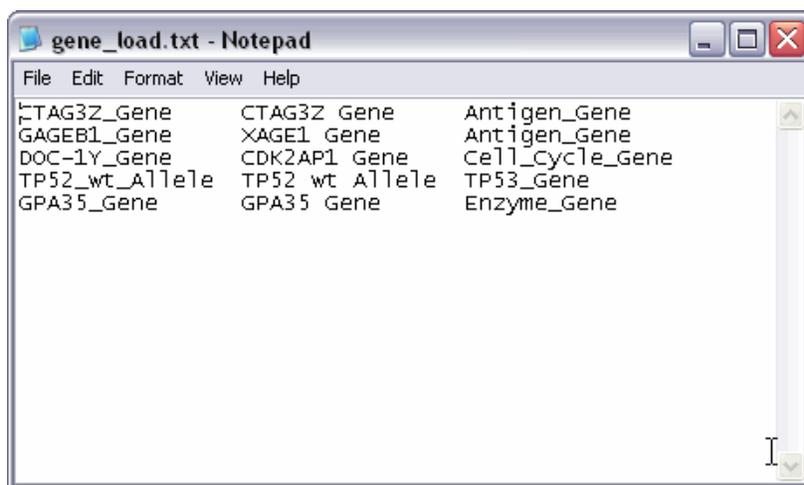


Figure 7.25 Tab-delimited input file for batch load

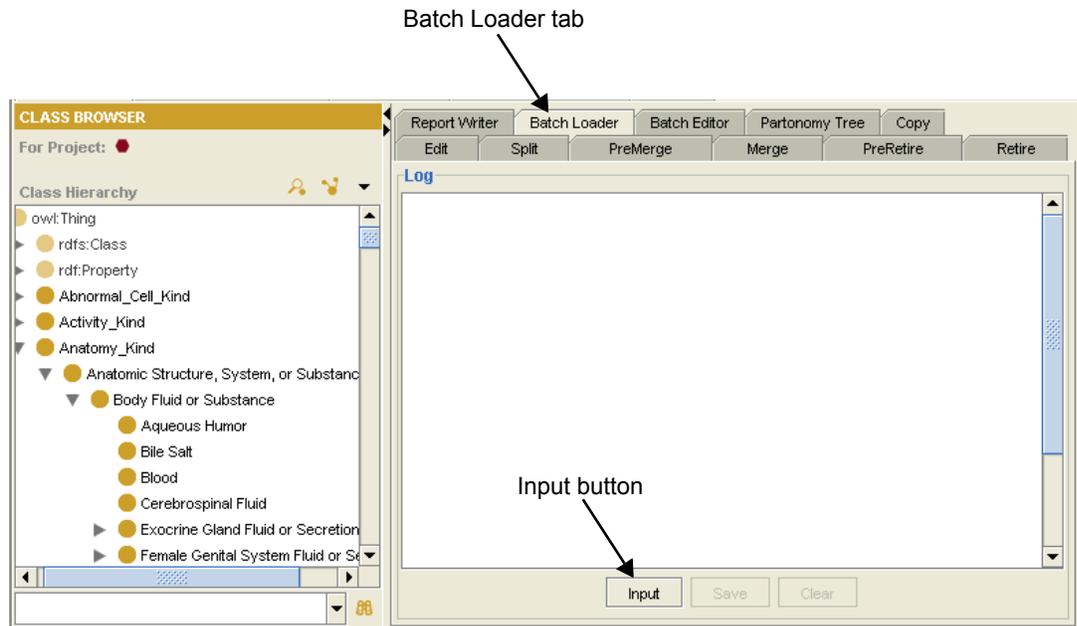
2. With the NCI Edit tab displayed, select any class in the Class Browser on the left (except *owl:Thing*).

**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 tab unless something other than *owl:Thing* is selected in the Class Browser.

3. Click the **Batch Loader** tab on the right.

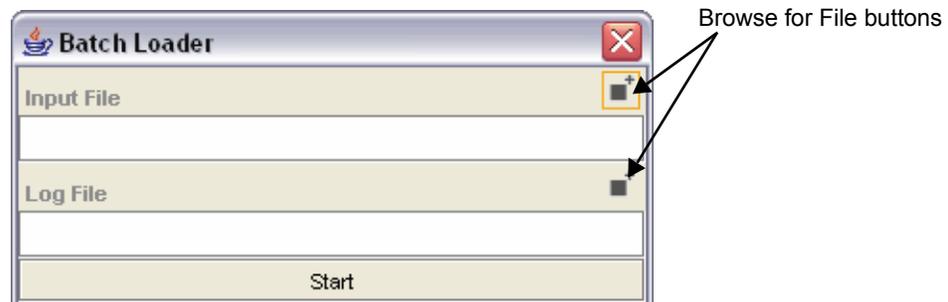
As shown in *Figure 7.26*, the tab is initially blank. Once you set up and run the batch load process, the display area of the tab shows the output of your batch load file.

4. Click the **Input** button at the bottom of the tab.



*Figure 7.26* Batch Loader tab

5. In the Batch Loader window (shown in *Figure 7.27*), 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 7.27* Batch Loader window

- b. In the Browse window, click the arrow to the right of the **Files of Type** field, then select **All Files** from the drop-down list.

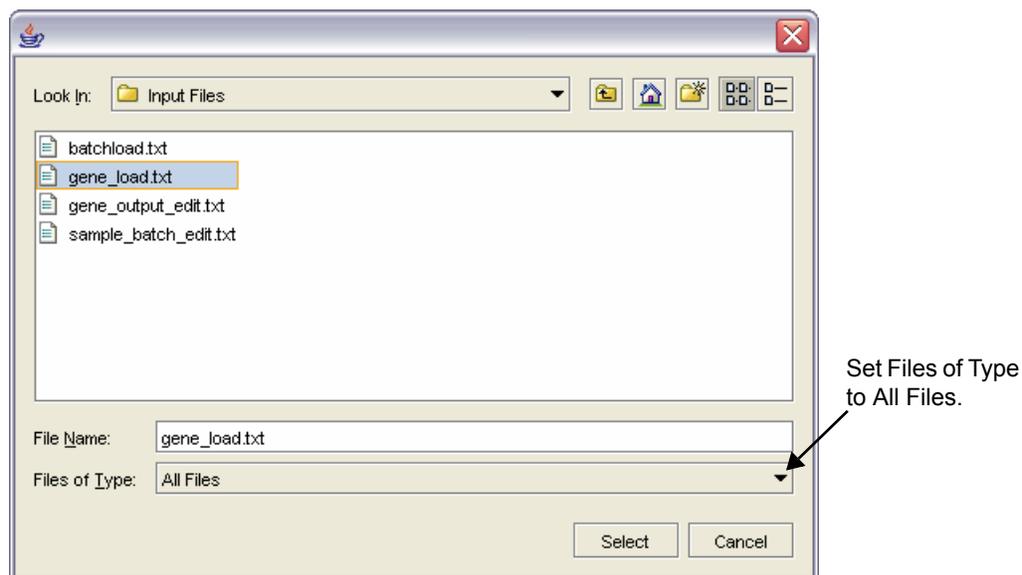


Figure 7.28 Browse window for batch load files

- 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 5. a. through 5. d. to browse for an output 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. Back in the Batch Loader window, ensure that both the **Input File** and **Log File** fields show a file path and name, as shown in [Figure 7.29](#).
  7. Click the **Start** button in the bottom of the window.

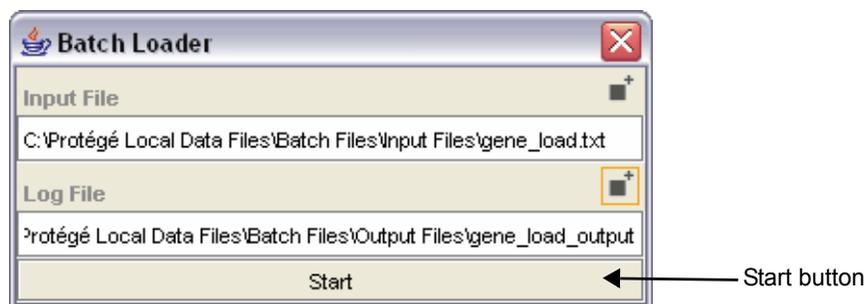
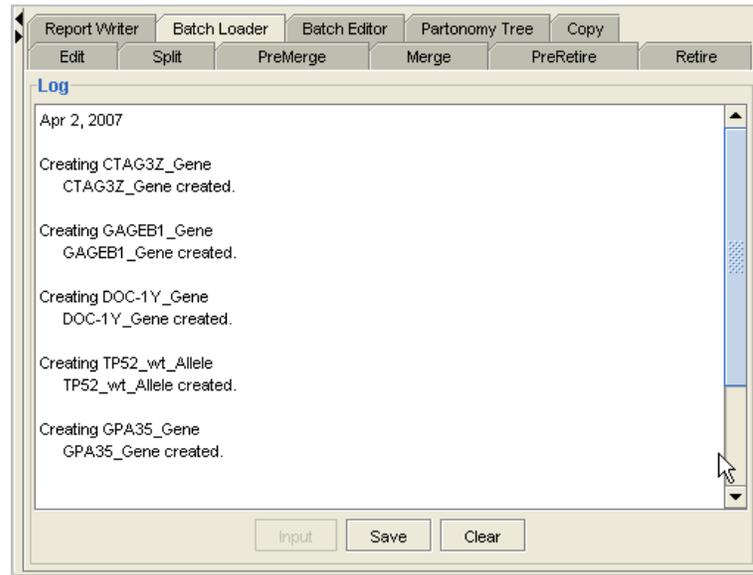


Figure 7.29 Batch Loader window with completed file paths and names

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. Click **Close** to dismiss the message window.

The names of all newly loaded classes appear in the Batch Loader display area, as shown in *Figure 7.30*.



*Figure 7.30 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 tab is still active, you do not need to click the button to save the new classes. They have already been saved to the knowledge base. You can, however, save the batch output to an external file as a record of the classes that you loaded.

---

## Editing a Batch of Classes

Batch editing is useful when you have a large number of classes and prefer to edit them outside of Protégé. You can then use the Batch Editor tab to import the edited classes into the application.

You can set up a batch edit file in a text editor such as Notepad, or you can set it up in a spreadsheet program such as Excel. In either case, save the file as a tab-delimited text file (.txt).

**Note:** For more information about working with .txt files in Excel, see the Excel online help.

The batch load file requires four to six fields, depending on what you are editing.

*Table 7.1* 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. <b>Note:</b> If the value in Field 3 is <i>parent</i> , the value in this field must be <b>new</b> or <b>delete</b> .
3	property   role   parent   association	Use one of these four accepted values to describe the property or parent class being modified. <b>Note:</b> If the value in this field is <i>parent</i> , the value in Field 2 must be <b>new</b> or <b>delete</b> .
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 <i>parent</i> . 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 7.1 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>
```

To edit 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 7.31](#). The file should be structured as described in [Table 7.1](#) on page 146.

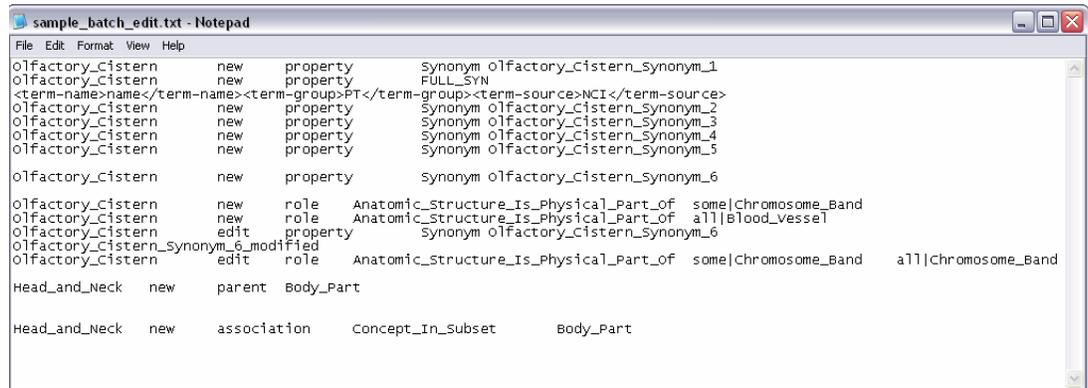


Figure 7.31 Tab-delimited input file for batch load

2. With the NCI Edit tab displayed, 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 tab unless something other than *owl:Thing* is selected in the Class Browser.

3. Click the **Batch Editor** tab on the right. As shown in [Figure 7.32](#), the tab is initially blank until you set up and run the batch edit process.
4. Click the **Input** button at the bottom of the tab.

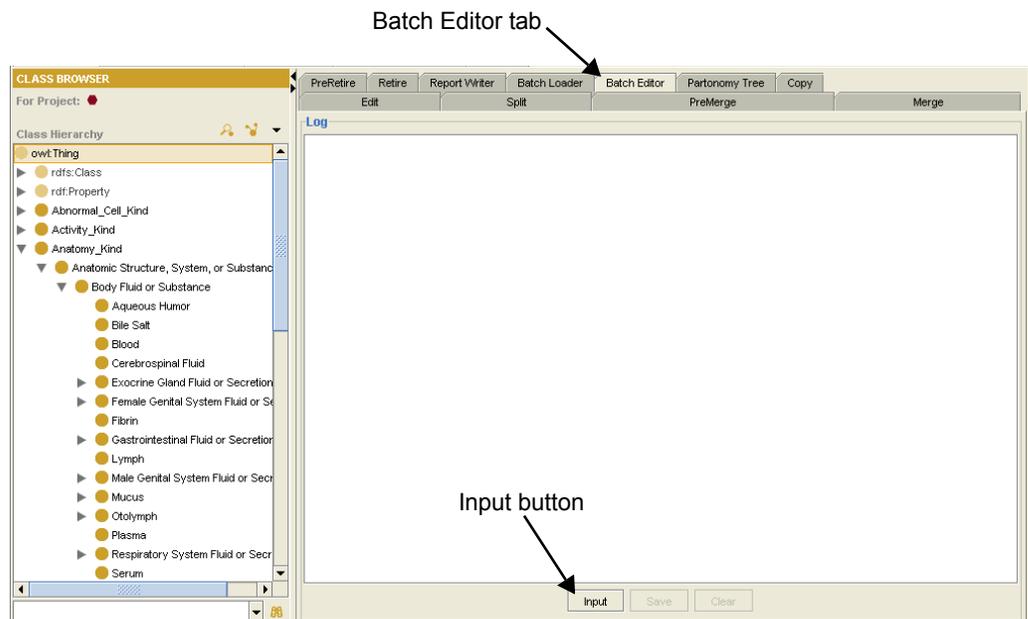


Figure 7.32 Batch Editor tab

5. In the Batch Editor window (shown in [Figure 7.33](#)), 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 .

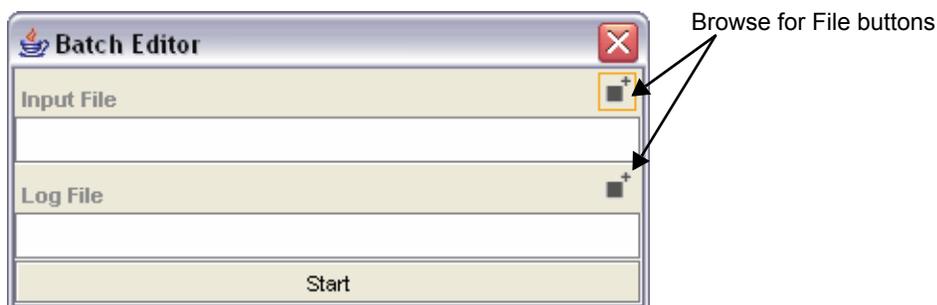


Figure 7.33 Batch Editor window

- b. In the Browse window, click the arrow to the right of the **Files of Type** field, then select **All Files** from the drop-down list.

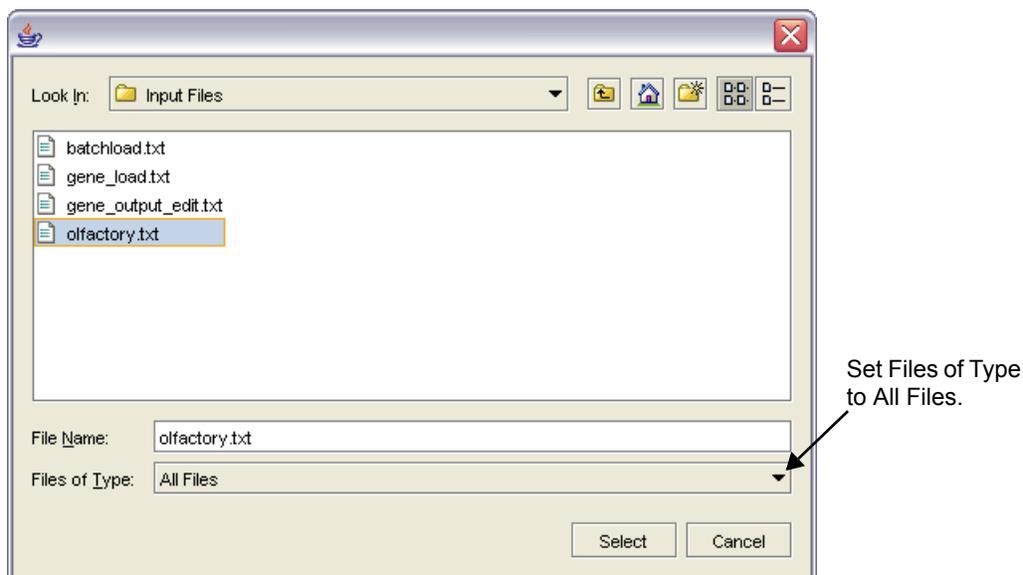


Figure 7.34 Browse window for batch edit files

- 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 [5. a.](#) through [5. d.](#) to browse for an output 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.

- Back in the Batch Editor window, ensure that both the **Input File** and **Log File** fields show a file path and name, as shown in *Figure 7.35*.

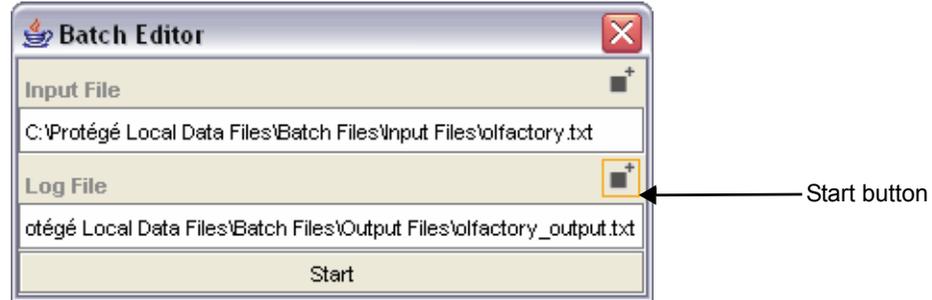


Figure 7.35 Batch Editor window with completed file paths and names

- Click the **Start** button in the bottom of the window. A progress bar confirms that the batch edit file load has begun. When the process is finished, a message window confirms the number of completed actions.
- Click **Close** to dismiss the message window. The names of all newly loaded classes appear in the Batch Loader display area, as shown in *Figure 7.36*.

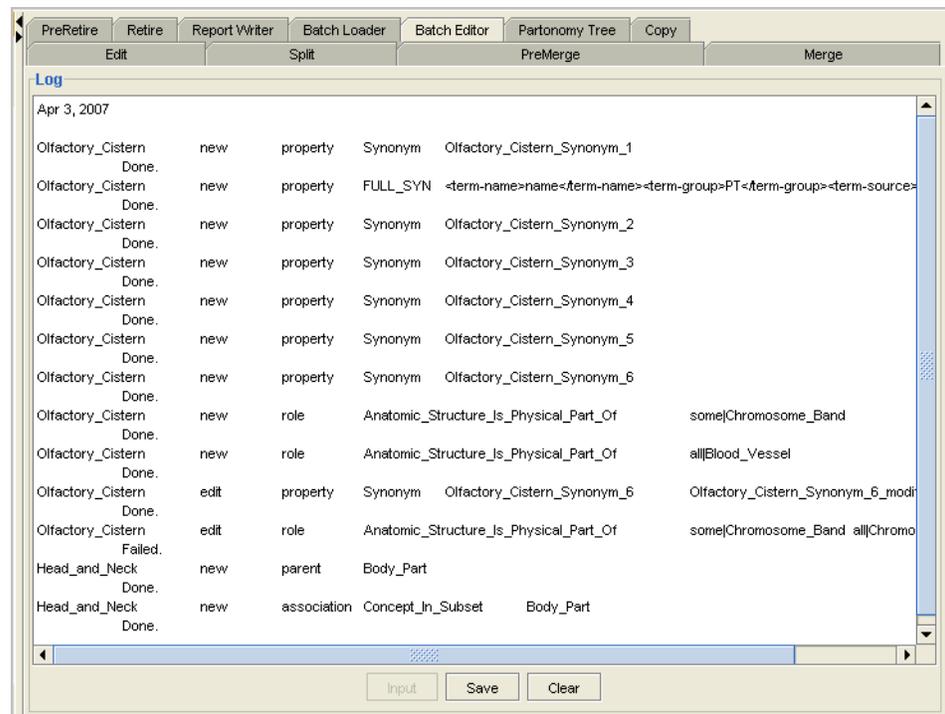


Figure 7.36 Newly imported batch edit

- 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 tab 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.

## Generating a Partonomy Tree

The **Partonomy Tree** tab enables you to select a root class and display it as a partonomy tree. A partonomy tree shows classes connected by *part\_of* relations.

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

1. Select a root class in the Class Browser on the left.
2. Click the **Partonomy Tree** tab on the right.
3. Click the **Tree** button on the bottom of the tab, as shown in [Figure 7.37](#).

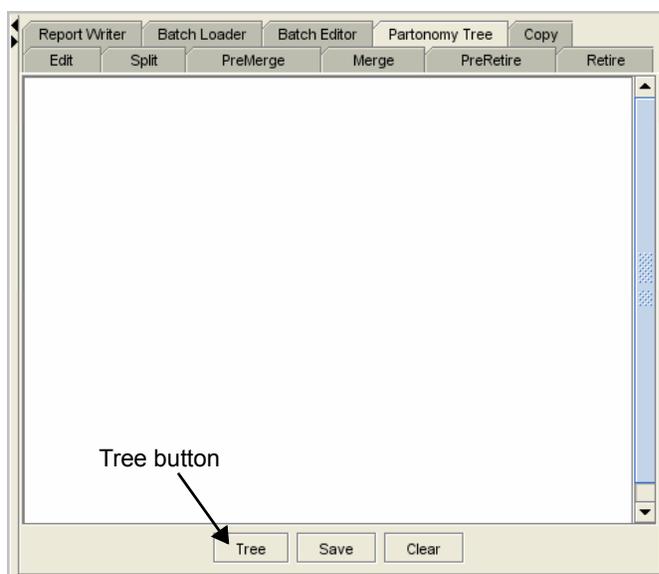


Figure 7.37 Partonomy Tree tab

4. In the Select Transitive Properties window (shown in [Figure 7.38](#)), select one or more restriction names. To select multiple restrictions, press and hold the CTRL key while clicking each item.

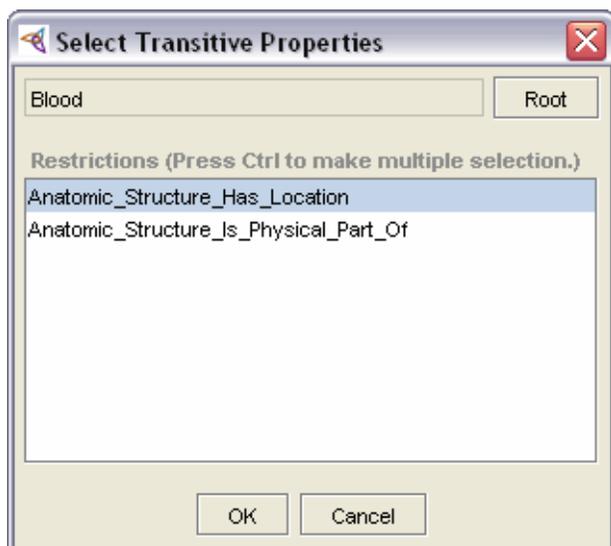


Figure 7.38 Select Transitive Properties window

5. Click **OK** to close the window and generate the tree. The partonomy tree appears in the tree view area of the tab, as shown in [Figure 7.39](#).

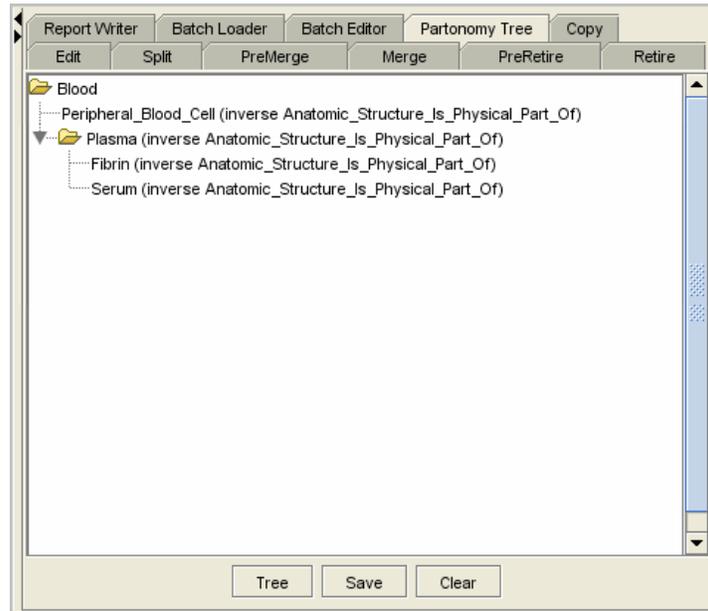


Figure 7.39 Partonomy tree for Blood class

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.

## Copying a Class

The **Copy** tab enables you to create a new class from an existing class. The original class essentially serves as a template for the new class. In Protégé parlance, this is known as *cloning*.

**Tip:** You can also copy properties and restrictions between cloned classes or between two different classes. To do this, simply drag one class into the upper pane, drag another class into the lower pane, then copy from one to the other.

To create a new class from an existing class, follow these steps:

1. Select the class to be copied in the Class Browser on the left. The **Edit** tab on the right displays basic properties for the selected class.
2. Click the **Copy** tab on the right. This tab has a split-pane view (upper and lower), similar to the **PreMerge** and **PreRetire** tabs.
3. Drag the selected class from the Class Browser into the upper pane of the **Copy** tab, as shown in *Figure 7.40*.
4. Copy the selected class by clicking the **Clone** button.

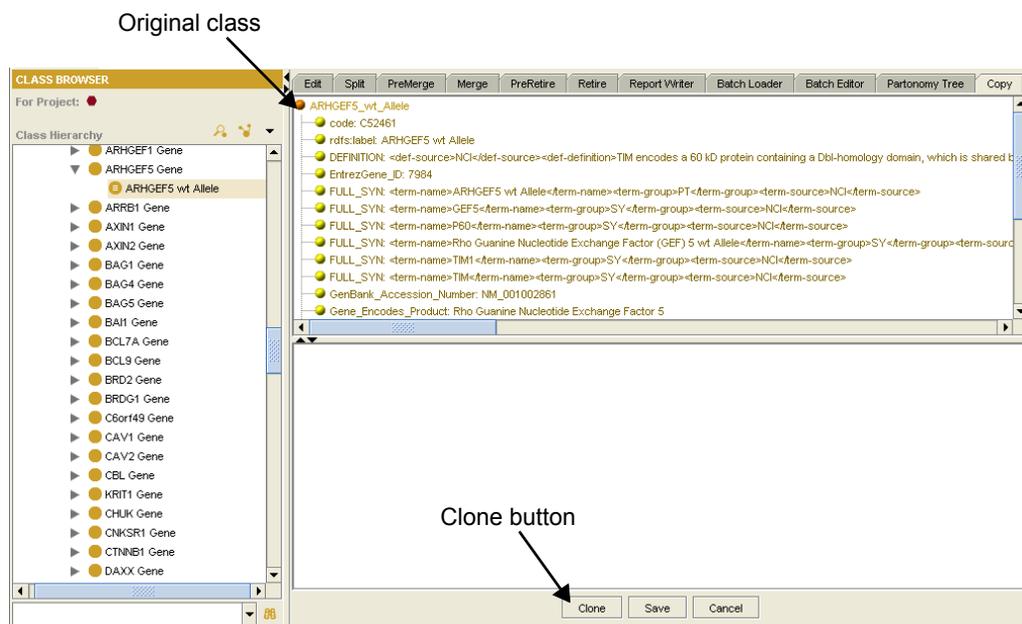


Figure 7.40 Copy tab with original class

- In the Enter Class Identifiers window, add the class name and preferred name, as shown in *Figure 7.41*.

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

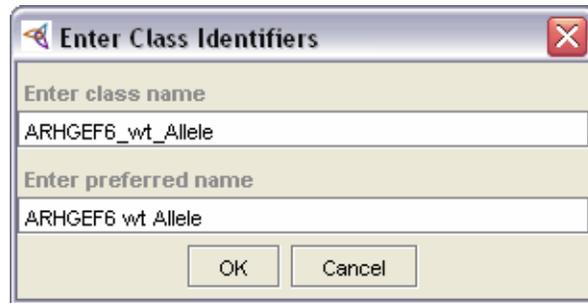


Figure 7.41 Enter Class Identifiers window

- Click **OK** to close the Enter Class Identifiers window. The new class now appears in the lower pane of the Copy tab, as shown in *Figure 7.42*.

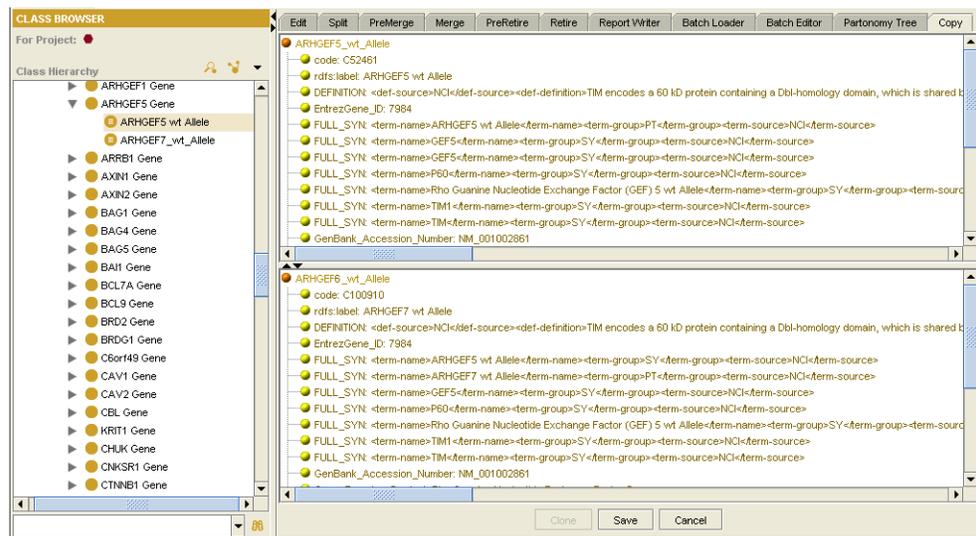


Figure 7.42 Copy tab with newly created class

### Tips for Editing in the Copy Tab Panes

The next steps in this procedure explain how to edit properties for either of the classes now displayed on the Copy tab, using a combination of left and right mouse clicks. When editing in the Copy tab panes (upper or lower), follow this sequence:

- Select a property or restriction with the **left** mouse button so that the item is highlighted.
- Right-click** on the same selected item, or right-click anywhere in the properties list. Both actions assume that you are editing the property that is highlighted.

**Note:** Right-clicking works only when you click on a selected item or in the property list. It has no effect when you click in the white space surrounding the list.

7. To modify properties for either class, follow these steps:
  - a. Select the property to be modified by clicking it with the **left** mouse button.
  - b. Right-click the selected item, then select the appropriate command from the shortcut menu (for example, **Modify Property**). See [Figure 7.43](#).

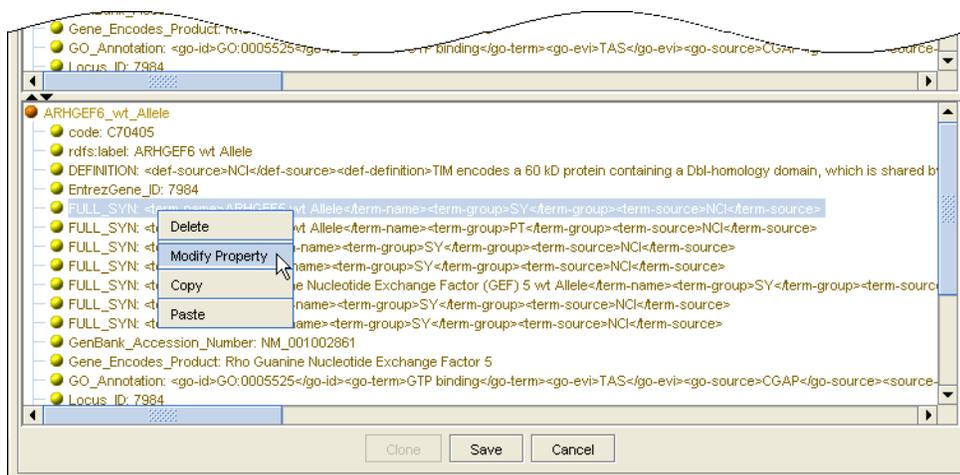


Figure 7.43 Right-click menu for modifying original and cloned classes

8. Click the **Save** button to accept the changes, then close any confirmation messages that appear.

In the Class Hierarchy, the new class now appears as a sibling of the original class.

## TDE vs. PROTÉGÉ TERMINOLOGY

*Table A.1* lists terms used in the Apelon Terminology Development Environment (TDE) and gives their equivalent terms in Protégé.

<i>TDE Term</i>	<i>Protégé Term or Description</i>
Top	Thing
Bottom	Nothing
Concept	<ul style="list-style-type: none"> <li>• Class (set semantics)</li> <li>• Individuals</li> </ul>
Kind	Top-level disjoint class
Role	<ul style="list-style-type: none"> <li>• ObjectProperty</li> <li>• DatatypeProperty</li> </ul>
Property	AnnotationProperty (AnnotationProperty of type DatatypeProperty)
Association	AnnotationProperty of type ObjectProperty
Role Expression	Restriction on property
All	allValuesFrom (all, only, universal quantifier)
Some	someValuesFrom (some, existential quantifier)
Qualifier	XML - as in TDE when we first began using it
(Pick Lists)	Enumerated Datatype AnnotationProperty
Primitive	Necessary conditions asserted
Defined	Necessary & Sufficient conditions asserted

*Table A.1 TDE terms and their Protégé equivalents*



# INDEX

## A

advanced queries  
  about 71 – 73  
  examples 74 – 83  
annotation properties  
  See properties  
anonymous classes 18  
associations, adding 119

## B

Basic Data subtab, about 46  
batch edit  
  procedure 146  
  tab interface, about 57  
batch load  
  procedure 142  
  tab interface, about 56

## C

Class Browser, about 43  
classes  
  about 89 – 90  
  adding pre-merge flag 127  
  adding pre-retire flag 131  
  copying 152  
  creating 94  
  merging 127, 130  
  retiring 131  
  splitting 124  
  treeing 98  
concepts in DL and OWL 18  
customizing search 84

## D

definitions  
  about 91 – 92  
  modifying 106  
description logic (DL)  
  about 16  
  concepts and roles 18

  in the NCI Thesaurus 23  
disjointness axiom 21  
downloading installation files 28  
downloading release notes 28

## E

Edit tab  
  about 45  
  subtabs  
    Basic Data 46  
    Properties 48  
    Relations 47  
Enterprise Vocabulary Services (EVS)  
  about 7  
  key terminologies 8  
  online resources 11  
  server hosting 9  
equivalency 21

## F

file format  
  for batch edit 146  
  for batch load 142  
FULL\_SYN property, about 91  
full synonym  
  adding  
  modifying 105

## I

input file  
  for batch editing 146  
  for batch load 142  
installing  
  application updates 31  
  main application file 27 – 30

## K

knowledge representation  
  first order predicate logic 14  
  frame-based 13

## L

local Thesaurus subset, using 35

logging in 32

## M

Merge tab, about 52

merging classes 127, 130

Metathesaurus

NCI

about 8

Web site 11

UMLS

about 10

Web site 12

## N

NCI Editor tab, about 42

NCI Extension, about 41

NCI Metathesaurus

about 8

Web site 11

NCI Thesaurus (NCIT)

about 8

semantic model for 13

Web site 11

necessary vs. necessary and sufficient 21

## O

output file

for batch editing 146

for batch load 142

for reports 136

OWL

about 16

anonymous classes 18

as used in NCI Thesaurus 24

class descriptions 18

concepts and roles 18

disjointness axiom 21

equivalency 21

existential and universal qualifiers 22

expression syntax (table) 24

language types 17

necessary vs. necessary and sufficient 21

property restrictions 22

subsumption axiom 21

## P

package file

See updating application

parent class

adding 99

deleting 102

modifying 101

partonomy tree

procedure 150

tab interface, about 58

pre-merge

procedure 127

tab interface, about 51

pre-retire

procedure 131

tab interface, about 53

project, setting up locally 38

properties

adding 104

definition, about 91 – 92

deleting 105

FULL\_SYN, about 91

modifying 105

Properties subtab, about 48

## Q

qualifiers

modifying 106

queries, advanced

about 71 – 73

examples 74 – 83

## R

relations

See restrictions

Relations subtab, about 47

release notes, downloading 28

report writer

procedure 136

tab interface, about 55

restrictions

adding

deleting 114

modifying 111

Retire tab, about 54

retiring a class 131

Review window 49, 93

right-clicking

on Copy tab 153

on PreMerge tab 128

on Split tab 125

role group, adding 115

roles in DL and OWL 18

## S

search

advanced queries

- about 71 – 73
  - examples 74 – 83
- customizing 84
- field and button 69
- results, reviewing 70
- simple, about 68
- simple vs. advanced 67
- Split tab, about 50
- splitting a class 124
- subclasses, re-treeing 132
- subsumption axiom 21
- superclasses
  - deleting 102
  - modifying 101
- support, contacting 5

## T

- tabs, about
  - Batch Editor 57
  - Batch Loader 56
  - Edit 45
  - Merge 52
  - NCI Editor 42
  - Partonomy Tree 58
  - PreMerge 51
  - PreRetire 53
  - Report Writer 55
  - Retire 54
  - Split 50
- TDE vs. Protégé terminology 155
- Thesaurus, NCI
  - about 8
  - semantic model for 13
  - Web site 11

## U

- UMLS Metathesaurus
  - about 10
  - Web site 12
- uninstalling previous versions 29
- updating Protégé application 31

## W

- Web Ontology Language
  - See OWL
- workflow manager tasks
  - merging flagged classes 130
  - retiring a class 135

## X

- XML tags
  - in advanced queries 72
  - in batch edit files 146

