NCI PROTÉGÉ 1.2

Editor's Guide

Center for Bioinformatics

This is a U.S. Government work.

May 29, 2008

CONTENTS

List of Figures vii

About This Guide 1

Purpose 1
Audience 1
Typical User 1
Prerequisites 2
Topics Covered 2
Additional References 3
Text Conventions Used 3
Credits and Resources 4
Application Support Contacts 5

Chapter 1
About the Enterprise Vocabulary Services 7

Introduction 7
Key EVS Terminologies 8
NCI Thesaurus 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
About Description Logics and OWL 15
Concepts and Roles 17
OWL Class Descriptions and Anonymous Classes 18
Subsumption and Disjointness Axioms 21
Equivalency and Necessary vs. Necessary and Sufficient Conditions 21
Property Restrictions and the Existential and Universal Qualifiers 22
Description Logic in the NCI Thesaurus 23
OWL and the NCI Thesaurus 24

Chapter 3
Getting Started with Protégé 27
Installing Protégé 27
Downloading the Installation File 28
Uninstalling Previous Versions of NCI Protégé 29
Installing the NCI Protégé Application 29
Logging In to the Protégé Server 30
Using a Local Thesaurus Subset 32
Downloading the Subset File 33
Installing the Subset File 33
Opening the Local Subset File in Protégé 33
Setting Up a Local Project 36

Chapter 4
About the NCI Protégé Work Environment 39
Overview of the Protégé Application 39
About NCI Protégé 39
About the Main Protégé Window 40
About the NCI Editor Tab 42
Class Browser 43
NCI Editor Subtabs 44
Working with the Console 46
Setting the Console Display Capacity 47
Capturing Information for Troubleshooting 48
Capturing a Thread Dump 48
Copying the Console Window Contents 48

This is a U.S. Government Work.

Submitting a Bug Report and File Attachment to GForge *49*

Chapter 5
Searching **53**

Simple vs. Advanced Search 53
Performing a Simple Search 54
Using the Simple Search Field and Button *55*
About the Search Results Display *56*
Selecting and Reviewing Search Results *56*
Building an Advanced Query 57
Advanced Query Syntax *59*
Tips for Using the Query-Building Interface *59*
Building a Simple Property Query: Preferred Name *61*
Building a Complex Property Query: FULL_SYN *62*
Building a Combined Property Query: Preferred_Name and FULL_SYN *64*
On Your Own... ..... *67*
Building a Restriction-Based Query *67*
Building an Association-Based Query *70*
Exporting Your Search Results 72
Building the Query *72*
Starting the Export *72*
Saving the File as an Excel Workbook *74*
Configuring the Advanced Query Tab 75
Changing the Default Search Property 75
Changing the Default Search Parameter 77
Verifying Search Configuration Changes 77
Understanding Where Search Properties and Values Are Stored 78
Available Data Types from SMI 79

Chapter 6
Working with Classes: The Basics **81**

About Classes 81
Class Identifiers 81
Class Relationships 82
Class Properties 82
About the FULL_SYN Property *82*
About the Definition Property *82*
Guidelines for Writing Definitions *83*
Examining Class Information 84
Basic Data Subtab *85*

Relations Subtab *86*
Properties Subtab *87*
Reviewing Changes As You Work 88
Creating a Class 89
Treeing a Class 92
Adding a Parent Class 93
Modifying a Parent Class 95
Deleting a Parent Class 96
Viewing Classes Connected by part_of Relations 97
About the Partonomy Tree Subtab 97
Generating a Partonomy Tree 98

Chapter 7
Working with Properties 101

Asserting Annotation Properties 101
Adding a Full Synonym 102
Modifying a Full Synonym 103
Deleting a Full Synonym 104
Modifying a Definition 104
Asserting Relations 106
Adding a Restriction 106
Modifying a Restriction 109
Deleting a Restriction 112
Adding a Role Group 113
Adding Multiple Restrictions *113*
Building a Role Group Expression *116*
Adding an Association 117
Editing Two Classes at the Same Time 120

Chapter 8
Splitting, Merging, and Cloning Classes 123

Splitting a Class 123
When to Consider Splitting a Class 123
About the Split Subtab 124
Using the Split Subtab to Split a Class 125
Examining the Properties of a Split Class 127
Merging Classes 127
When to Consider Merging Classes 127
About the PreMerge Subtab 128
Using the PreMerge Subtab to Flag Classes for a Merge 129
Finding the Retiring Class under the Premerged_Concepts Branch ... 131

This is a U.S. Government Work.

About the Merge Subtab 131
Using the Merge Subtab to Merge Flagged Classes 132
Cloning a Class 133

Chapter 9
Pre-Retiring and Retiring Classes 137

About the Retirement Process 137
When to Consider Retiring a Class 137
PreRetire vs. Retire 137
Pre-Retiring a Class 138
About the PreRetire Subtab 138
Using the PreRetire Subtab to Flag a Class for Retirement 139
Selecting a Class for Retirement 139
Eliminating Subclass Dependencies 139
Saving the Class with a Pre-Retired Status 143
Finding the Pre-Retired Class in the Hierarchy 143
Retiring a Pre-Retired Class 144
About the Retire Subtab 144
Using the Merge Subtab to Retire a Flagged Class 145

Chapter 10
Using NCI Protégé Utilities 147

Generating Reports for Selected Classes 147
About the Report Writer Subtab 147
Using the Report Writer Subtab to Generate Reports 148
Creating an Output Directory and Files 148
Generating a Report 149
Loading a Batch of Classes for Editing 154
About the Input File Format for a Batch Load 154
About the Batch Loader Subtab 154
Using the Batch Loader Subtab to Load an Input File 155
Loading a Batch of Edited Classes 159
About the Input File Format for a Batch Edit 159
About the Batch Editor Subtab 160
Using the Batch Editor Subtab to Import a Batch of Edited Classes 161

Chapter 11
Workflow Procedures for NCI Editors 165

Overview of BiomedGT 165
About BiomedGT 165
Purpose 165

Audience *166*
About the BiomedGT Wiki 166
Overview of Internal NCI Workflow 168
Primary Goals 168
Workflow Roles 168
Workflow Stages 169
About the NCI Workflow Tab 170
Workflow Procedures 172
Accepting a Workflow Assignment 172
Editing an Assigned Concept 173
Adding Notes to an Assignment 174
Using the NCI Workflow Tab *174*
Using the Workflow Subtab *174*
Rejecting an Assignment 175
Completing an Assignment 176
Suggesting an Assignment 176

Appendix A
TDE vs. Protégé Terminology 177

Index 179

This is a U.S. Government Work.

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. Winzip Self-Extractor window - new installation 29
Figure 3.2. Welcome to Protégé window 30
Figure 3.3. Open Project window - server button 31
Figure 3.4. Select Project window 31
Figure 3.5. Main NCI Protégé window 32
Figure 3.6. Welcome to Protégé window 34
Figure 3.7. Open Project window - File button 34
Figure 3.8. Protégé standard interface with no visible NCI-specific tabs 35
Figure 3.9. Configure File window with ordered tab widgets 36
Figure 3.10. New local configuration 37
Figure 4.1. Main NCI Protégé window 40
Figure 4.2. NCI Editor tab with Class Browser and subtabs 42
Figure 4.3. Class Browser 43
Figure 4.4. Console window 46
Figure 4.5. Console window menu - Properties command 47
Figure 4.6. Console Properties window 47
Figure 4.7. Command Line window menu - Select All command 48
Figure 4.8. Tracker page - Submit New link 49
Figure 5.1. Simple search field and button 53
Figure 5.2. NCI Editor tab with simple search field and button 54
Figure 5.3. Select Property window with simple search field and button 54
Figure 5.4. Simple search field and Search for Class button 55
Figure 5.5. Results for simple search 55
Figure 5.6. Results: Preferred_Name ends with fluid 56
Figure 5.7. Advanced Query tab 57
Figure 5.8. Default query 59
Figure 5.9. Parameter list 60
Figure 5.10. Default query 61
Figure 5.11. Default query 62
Figure 5.12. Select Slot window 62
Figure 5.13. Parameter list - starts with 63

Figure 5.14. Finished query - FULL_SYN starts with *>benign 63
Figure 5.15. Default query 64
Figure 5.16. Selecting from parameter list 64
Figure 5.17. Completed query - Preferred_Name starts with bcl*gene 64
Figure 5.18. Select Slot window 65
Figure 5.19. Completed query - FULL_SYN starts with *>bcl*gene 65
Figure 5.20. Combined queries 66
Figure 5.21. Full query with search results 66
Figure 5.22. Removing the default query 67
Figure 5.23. Initial setup for restriction-based query 68
Figure 5.24. Select property window 68
Figure 5.25. Restriction-based query with search property 69
Figure 5.26. Restriction-based query with completed fields 69
Figure 5.27. Search results for restriction-based query 70
Figure 5.28. Removing the default query 70
Figure 5.29. Association-based query with search property 71
Figure 5.30. Export configuration window 72
Figure 5.31. Export configuration window with specified settings 74
Figure 5.32. Default query with new search property and search parameter 75
Figure 5.33. Configure File window with protege.properties tab selected 76
Figure 5.34. Configure File window - Property Files tab with editable field 76
Figure 5.35. Configure File window with new search property and parameter 77
Figure 6.1. Edit subtab 84
Figure 6.2. Basic Data subtab 85
Figure 6.3. Relations subtab 86
Figure 6.4. Properties subtab 87
Figure 6.5. Review window 88
Figure 6.6. Creating a new class 89
Figure 6.7. Create Subclass window 90
Figure 6.8. Select a superclass window 90
Figure 6.9. Create Subclass window with completed fields 91
Figure 6.10. New class information 92
Figure 6.11. Edit subtab - Parent Class panel 93
Figure 6.12. Select a class window 94
Figure 6.13. Parent Class panel with new class 94
Figure 6.14. Modify Named Superclass window 95
Figure 6.15. Select a named class window 95
Figure 6.16. Modified parent class 96
Figure 6.17. Partonomy Tree subtab 97
Figure 6.18. Partonomy Tree subtab 98
Figure 6.19. Select Transitive Properties window 98
Figure 6.20. Partonomy tree for Blood class 99
Figure 7.1. Create FULL_SYN Annotation Property window 102
Figure 7.2. New synonym 103
Figure 7.3. Edit FULL_SYN Annotation Property window 103
Figure 7.4. Edit DEFINITION Annotation Property window 105
Figure 7.5. Create a Restriction window (initial) 106

Figure 7.6. Create a Restriction window (second of two windows) 107
Figure 7.7. Select a named class window 107
Figure 7.8. Create a Restriction window with Filler value 108
Figure 7.9. Newly added restriction 108
Figure 7.10. Restrictions/Groups panel with inherited restrictions 109
Figure 7.11. Edit a Restriction window 110
Figure 7.12. Modify a Restriction window 110
Figure 7.13. Select a named class window 111
Figure 7.14. Modify a Restriction window with new Filler value 111
Figure 7.15. Newly modified restriction 112
Figure 7.16. Edit sub > Relations subtab > Restrictions panel 113
Figure 7.17. Create a Restriction window (first of two windows) 113
Figure 7.18. Create a Restriction window (second of two windows) 114
Figure 7.19. Select a named class window 114
Figure 7.20. Create a Restriction window with Filler value 115
Figure 7.21. Multiple restrictions 115
Figure 7.22. Role group example 116
Figure 7.23. Role group expression 116
Figure 7.24. Relations subtab for Phagocytosis 117
Figure 7.25. Add an Object-Valued Property window 118
Figure 7.26. Select a Property window 118
Figure 7.27. Select a property value added 119
Figure 7.28. Add an Object-Valued Property window with stored values 119
Figure 7.29. Associations panel 119
Figure 8.1. Split subtab 124
Figure 8.2. Existing Concept pane 125
Figure 8.3. Enter Class Identifiers window 125
Figure 8.4. Split classes 126
Figure 8.5. Properties of new class 127
Figure 8.6. PreMerge subtab 128
Figure 8.7. PreMerge subtab with surviving and retiring concepts 129
Figure 8.8. Enter Notes window 130
Figure 8.9. Merge subtab 131
Figure 8.10. Merging two classes 132
Figure 8.11. Copy subtab with original class 133
Figure 8.12. Enter Class Identifiers window 134
Figure 8.13. Copy subtab with newly created class 134
Figure 9.1. PreRetire subtab 138
Figure 9.2. Hiding the Class Browser 139
Figure 9.3. Selected subclass with Basic Data subtab displayed 140
Figure 9.4. Relations subtab with one parent class 140
Figure 9.5. Select a class window 141
Figure 9.6. Buttons sets for inner, nested subtabs and outer subtab 142
Figure 9.7. Enter Notes window 143
Figure 9.8. Retire subtab 144
Figure 10.1. Report Writer subtab and Report Writer window 148
Figure 10.2. Report Writer subtab - Generate button 149

Figure 10.3. Report Writer window for specifying an output file 150
Figure 10.4. File browser window 150
Figure 10.5. Level numbers for Exocrine Gland Fluid or Secretion 151
Figure 10.6. Report Writer window with completed values 152
Figure 10.7. Report Writer status window as it initially appears 152
Figure 10.8. Report with classes only - no attributes 153
Figure 10.9. Report with classes, subclasses, properties, and restrictions 153
Figure 10.10. Batch Loader subtab 154
Figure 10.11. Tab-delimited input file for batch load 155
Figure 10.12. Batch Loader subtab 156
Figure 10.13. Batch Loader window 156
Figure 10.14. Browse window for batch load files 157
Figure 10.15. Batch Loader window with completed file paths and names 157
Figure 10.16. Newly imported batch load 158
Figure 10.17. Batch Editor subtab 160
Figure 10.18. Tab-delimited input file for batch load 161
Figure 10.19. Batch Editor subtab 161
Figure 10.20. Batch Editor window 162
Figure 10.21. Browse window for batch edit files 162
Figure 10.22. Batch Editor window with completed file paths and names 163
Figure 10.23. Newly imported batch edit 163
Figure 11.1. BiomedGT Home page 166
Figure 11.2. NCI Workflow tab 170
Figure 11.3. Workflow items with ASSIGNED status 172
Figure 11.4. Assignment from wiki import - proposed and existing concepts 173
Figure 11.5. Window for adding notes to an assignment 174
Figure 11.6. Message confirming creation of suggested assignment 176

About This Guide

This section introduces you to the *NCI Protégé 1.2 Editor's Guide*.

Topics in this Section

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

Purpose

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

Audience

Typical User

This guide is designed for the following users:

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

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

Prerequisites

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

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

Topics Covered

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

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

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

Additional References

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

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

Text Conventions Used

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

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

Credits and Resources

The following people contributed to the development of this guide.

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

Application Support Contacts

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

Support Option	*Additional Information*
For NCI Protégé Users	
GForge	If you are an NCI Protégé user, post all bug reports and requests for enhancements on GForge using the following URL: https://gforge.nci.nih.gov/tracker/?group_id=174 For instructions on reporting bugs, see *Submitting a Bug Report and File Attachment to GForge* on page 49.
Administrative tasks for NCI Protégé	EVS Operations Support can provide the following services: • Loading a database • Creating a user account • Creating or configuring a Protégé project • Installing or reinstalling the Protégé client on a user's machine **Note:** If you need any of these services, send your request to the e-mail address under the **For General Users** section (below).
For General Users	
E-mail	Write to ncicb@pop.nci.nih.gov and include the following information: • Your contact information, including your phone number • The name of the application that you are using • The URL (for Web-based applications) • A description of the problem and the steps required to recreate it • The text of any error messages you have received.
Web	NCICB support: http://ncicbsupport.nci.nih.gov/sw/ NCI Protégé online support forums: https://gforge.nci.nih.gov/forum/?group_id=174
Telephone	Local: 301-451-4384 Toll-free: 888-478-4423 Telephone support is available Monday through Friday, 8 a.m. - 8 p.m. Eastern time, excluding government holidays.

Chapter 1

About the Enterprise Vocabulary Services

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

Topics in this Chapter

- *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

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

To respond to the need for consistency among various NCI projects and initiatives, the NCI publishes the NCI Thesaurus and the NCI Metathesaurus. The caCORE EVS interfaces, discussed later in this guide, provide access to both of these terminologies, which are discussed in the next two sections.

NCI Thesaurus

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

The NCI Thesaurus 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 Thesaurus 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. Additional external, proprietary vocabularies include MedDRA, SNOMED, and ICD-O-3, among others.

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

Table 1.1 NCI local source vocabularies included in the Metathesaurus

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

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

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

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 the Health Level Seven (HL7) organization 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 server, 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 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, a concept may sometimes 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 can 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*.

Relationship	***Description***
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.

Resource	***Location***
EVS main Web site	http://evs.nci.nih.gov
NCI Thesaurus (NCIT) and NCI Terminology Browser	http://nciterms.nci.nih.gov/NCIBrowser/Dictionary.do
NCI Metathesaurus	http://ncimeta.nci.nih.gov/MetaServlet/
NCI Distributed Terminology Server	http://nciterms.nci.nih.gov/NCIBrowser/Dictionary.do

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.

Resource	***Location***
caCORE Technical Guide	http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview
caCORE API	http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/caBIO
EVS Technical Overview	http://evs.nci.nih.gov/moreInformation/
EVS Production and Server Environment	http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary/implementation

Table 1.4 EVS online resources (Technical)

Other

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

Resource	***Location***
FDA Terminology Data Standards	http://www.fda.gov/oc/datacouncil/spl.html
UMLS Metathesaurus	http://www.nlm.nih.gov/research/umls/umlsdoc.html

Table 1.5 Other online resources

From Here...

- *Description Logic and the NCI Thesaurus Semantic Model* on page 13
- *Getting Started with Protégé* on page 27

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.

Topics in this Chapter

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

About Knowledge Representation

Frame-Based Representations

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

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

In a seminal paper by Marvin Minsky published in 1975, the author placed the frame representation paradigm in the context of a semantic network of nodes, attributes, and

relations. *Figure 2.1* shows a simple frame-based representation of an earthquake as it might be used in a semantic network of news stories.[1]

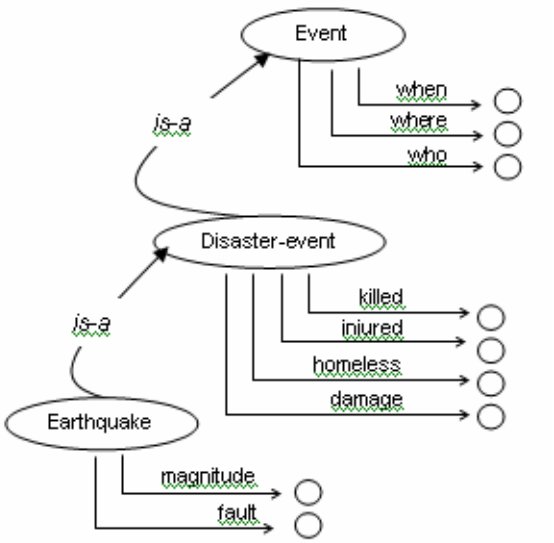


Figure 2.1 An Earthquake in a semantic network of news stories

First-Order Predicate Logic

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

Propositional Logic	***First-Order Predicate Logic***
All men are mortal. Socrates is a man. Socrates is mortal.	$\forall x$: Man (x) => Mortal(x) Man(Socrates) => Mortal(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 to make this information explicit, however, soon revealed that the problem was computationally intractable. This occurred for two reasons: (1) The frame representation was too permissive; more rigorous definitions were required to make the representation computational; and (2) first-order predicate logic itself was computationally intractable.

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

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

About Description Logics and OWL

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

Today, there is a large family of description logics that have been realized with varying levels of expressivity and resulting computation complexities (*Table 2.2* on page 16). The difference among DLs boils down to the construction operations allowed by the specific type of DL. For example, the most minimal form of a DL does not allow any form of existential quantification. This limitation allows for a very easily computed

solution space, but the resulting expressivity is severely diminished. Reasoners (classifiers) can be designed to deal only with the specific constructors of that DL, therefore simplifying computational effort. The theory behind these models is beyond the scope of this discussion. If you are interested in further exploring this theory, read *The Description Logic Handbook* by Franz Baader, et al. (Eds.), Cambridge University Press, 2003, ISBN 0-521-78176-0.

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

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

Table 2.2 DL expressivities

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

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

OWL has three levels of expressivity:

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

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

Concepts and Roles

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

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

A subtle distinction exists between a DL concept and a terminologic concept: the 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 NCI Protégé; however, some terms might show vestiges of the previous editing environment during a transition period.

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

OWL Class Descriptions and Anonymous Classes

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

OWL provides six different concept construction/description mechanisms:

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

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

some Gene_Plays_Role_In_Process_Signal_Transduction

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

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

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

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

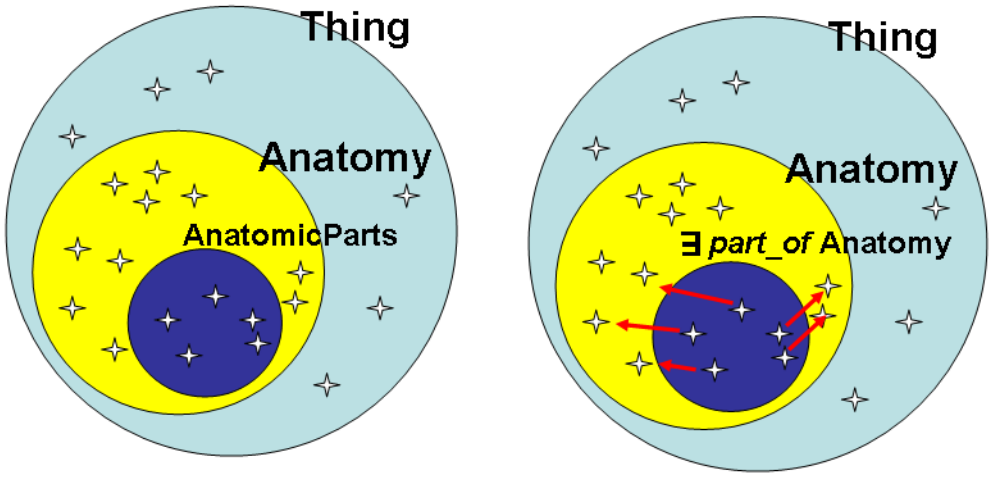


Figure 2.2 "Thing" OWL class and some subtypes of the anatomic domain

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

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

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

sick = complementOf(healthy)

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

Suppose, for example, that the narrative of an experimental observation or a clinical finding includes the description that "the liver is not affected." Further, the terminology of the anatomy domain is constructed such that the parts of liver are identified, and the editor intends to include parts of liver in the formal description or definition of the

observation-concept or finding-concept so that end users can code artifacts with it. The concept will likely have additional conditions asserted; however, for the present argument we need only consider the "parts of liver" portion.

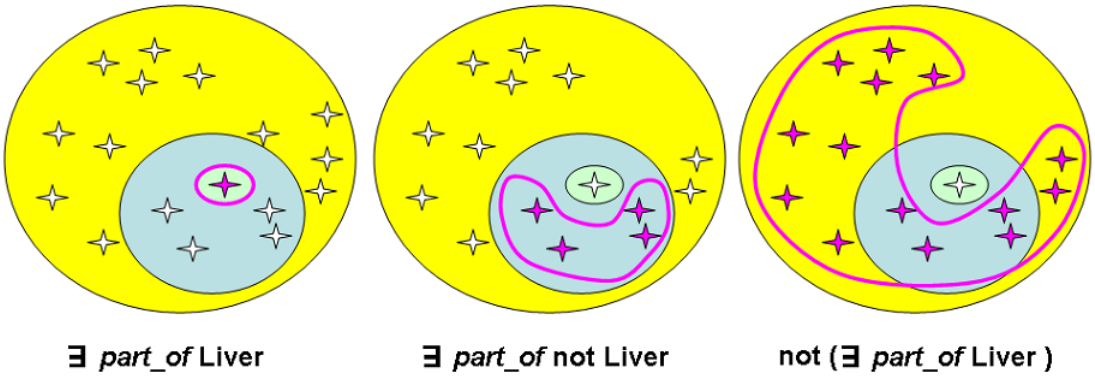


Figure 2.3 Effect of complement constructor on anonymous classes described by part_of role with domain anatomicpart and range anatomy

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

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

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

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

Subsumption and Disjointness Axioms

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

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

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

Equivalency and Necessary vs. Necessary and Sufficient Conditions

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

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

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

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

In simple terms, when a class such as A is *primitive*, class A can ask of its subclasses B, and C “do you meet my conditions for membership?” In contrast, when a class is

defined, such as C, C can ask this same question from E, but E can also ask from C "do I meet your conditions for membership?" However, this doesn't apply only to E. For instance, D could ask the same question from C, and if D met C's conditions, it would be inferred that D is a subclass of C as well.

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

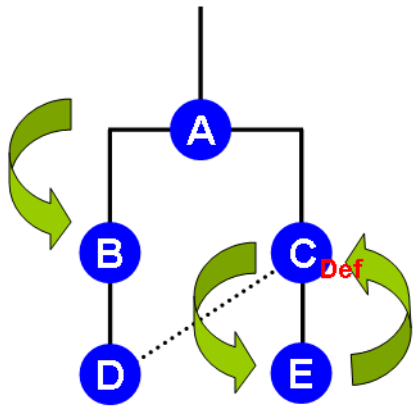


Figure 2.4 Hierarchy of named classes A, B, C, D, E, with Class C as defined

Property Restrictions and the Existential and Universal Qualifiers

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

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

some Disease_Has_Finding Amorphous_Eosinophilic_and_Acellular_Deposit

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

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

all Disease_Has_Finding Amorphous_Eosinophilic_and_Acellular_Deposit

says that **if** the property *Disease_Has_Finding* exists in an individual of this class, it can only have the value *Amorphous_Eosinophilic_and_Acellular_Deposit*. (The property need not be specified for an individual member of the class unless this condition is

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

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

Description Logic in the NCI Thesaurus

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

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

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

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

OWL Element	***Symbol***	***Key***	***Example***
allValuesFrom	$\forall$	*	$\forall$ children Male
someValuesFrom	$\exists$	?	$\exists$ children Lawyer
hasValue	$\square$	$	rich $\square$ true
cardinality	=	=	children = 3
minCardinality	$\geq$	>	children $\geq$ 3
maxCardinality	$\leq$	<	children $\leq$ 3
complementOf	$\neg$	!	$\neg$ Parent
intersectionOf	$\cap$	&	Human $\cap$ Male

Table 2.3 Owl expression syntax used in Protégé

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

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

Table 2.3 Owl expression syntax used in Protégé (Continued)

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

OWL and the NCI Thesaurus

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

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

Identifiers

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

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

Class Expressions

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

Property Hierarchy

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

Property Inverses

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

Datatype Properties

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

Transitivity of Properties

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

More Precise Declaration of Property Domains and Ranges

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

Some vs. All Property Restrictions

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

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

Defining and Non-Defining Restrictions

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

Disease-Anatomy Relationships

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

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

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

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

Property Qualifiers

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

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

Associations

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

Exceptional Conditions

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

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

From Here...

- *Getting Started with Protégé* on page 27
- *About the NCI Protégé Work Environment* on page 39
- *Searching* on page 53

CHAPTER

3

GETTING STARTED WITH PROTÉGÉ

This chapter explains how to download the Protégé application files, install the application, and log in. It also includes instructions for installing a local NCI Thesaurus subset file in case you want to work with NCI Protégé offline.

Topics in this Chapter

- *Installing Protégé* on this page
- *Logging In to the Protégé Server* on page 30
- *Using a Local Thesaurus Subset* on page 32

Installing Protégé

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

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

Category	***Subcategory/File name***	***Purpose***
Base Installation	Production Client Build: `ProtegeClient-1.2.x.exe` (Windows users) `ProtegeClient-1.2.x.zip` (Mac users)	Installs the Protégé application. The `.exe` version unzips automatically when you run it on a Windows machine, but it does not run on a Mac. If you are a Mac user, download and install the `.zip` version.

Table 3.1 Installation files for Protégé and the NCI Extension

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

Table 3.1 Installation files for Protégé and the NCI Extension (Continued)

Downloading the Installation File

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

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

1. Log in to GForge at http://gforge.nci.nih.gov/.
2. Locate the **NCICB EVS Collaborative Terminology Development Tools** project at http://gforge.nci.nih.gov/projects/protegegui/.
3. Click the **Files** tab.
4. Select one of the following files under the **Base Installation** category:
   - `ProtegeClient-1.2.x.exe` (Windows users), or
   - `ProtegeClient-1.2.x.zip` (Mac users)
5. Save the file to a temporary directory on your hard drive.
6. Note the name of the directory in which you stored the files.

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

Uninstalling Previous Versions of NCI Protégé

If you have installed a previous version of NCI Protégé, remove the previous version before installing a new version. To uninstall a previous version, follow these steps:

1. Using Windows Explorer, locate the following directory:

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

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

2. Delete the entire directory.

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

Installing the NCI Protégé Application

Prerequisite task: *Downloading the Installation File* on page 28

To install the NCI Protégé application, follow these steps:

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

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

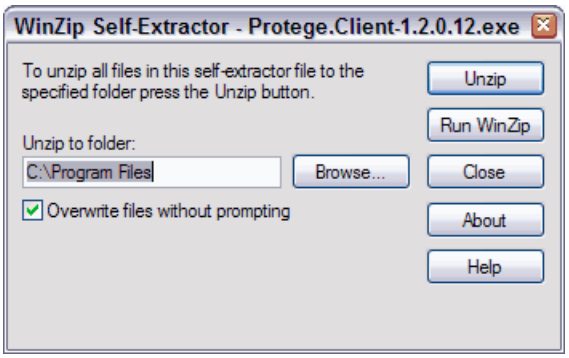


Figure 3.1 Winzip Self-Extractor window - new installation

2. Ensure that the **Overwrite files without prompting** box is checked, then click **Unzip**.
3. When the **WinZip Self-Extractor** message window appears, click **OK** to close the message window.
4. Click **Close** to close the **WinZip Self-Extractor** window.
5. Using Windows Explorer, browse to confirm that the following directory was created on your hard drive:

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

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

Logging In to the Protégé Server

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

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

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

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

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

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

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

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

3. In the **Welcome to Protégé** window, select **Open Existing File** (*Figure 3.2*).

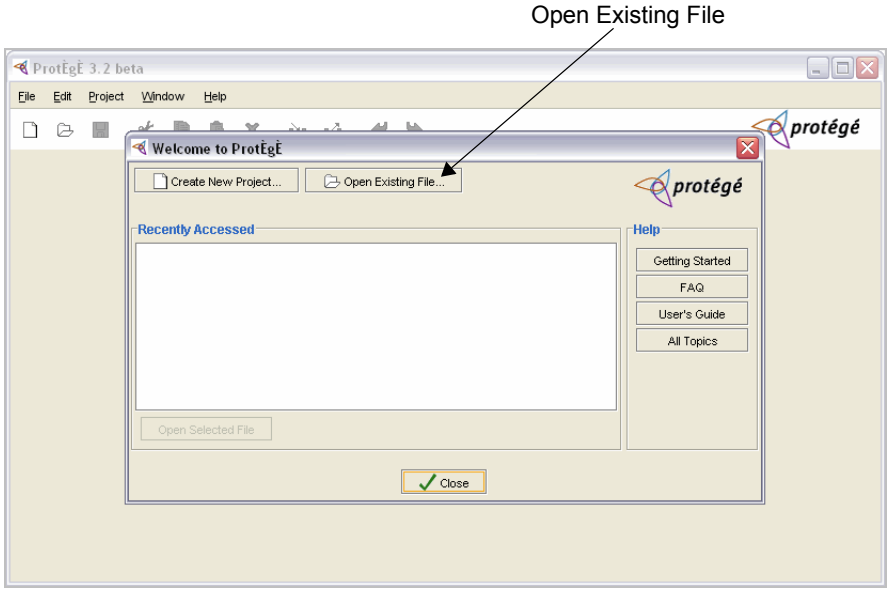


Figure 3.2 Welcome to Protégé window

4. In the **Open Project** window, click **Server** if it is not already selected (*Figure 3.3*).

5. Enter the **Host Machine Name** if necessary.

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

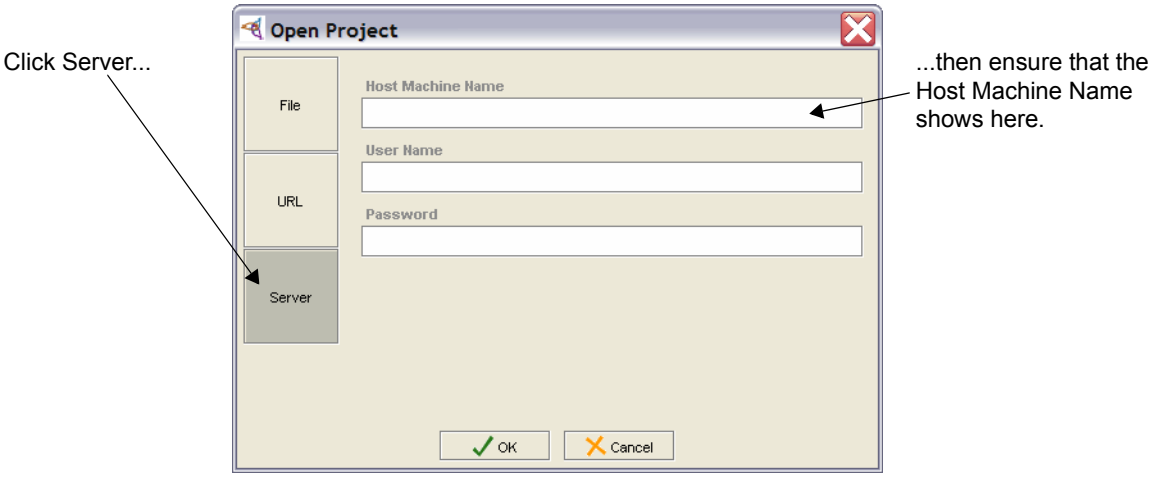


Figure 3.3 Open Project window - server button

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

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

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

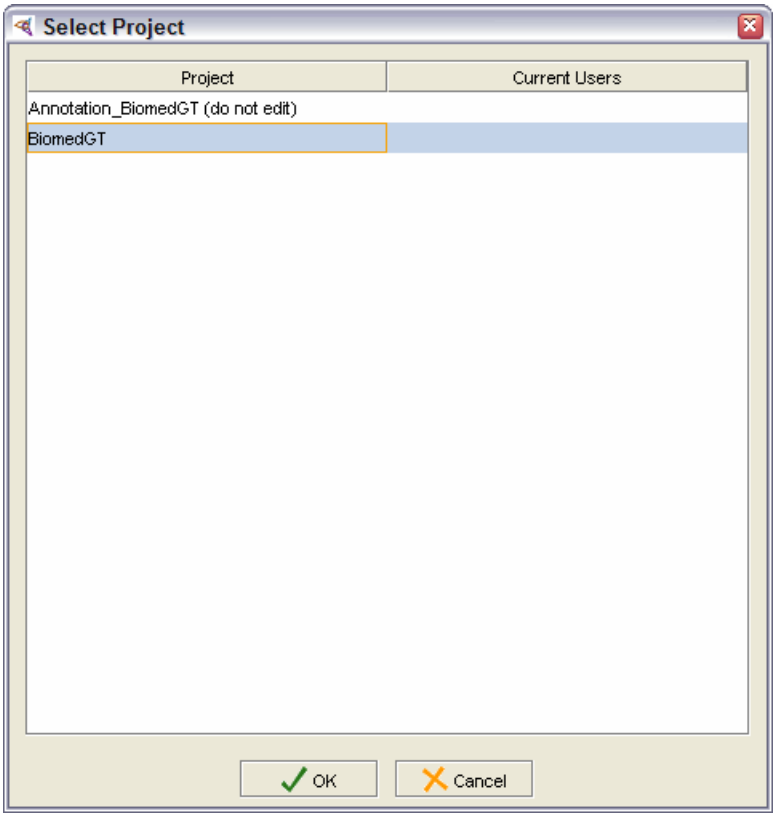


Figure 3.4 Select Project window

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

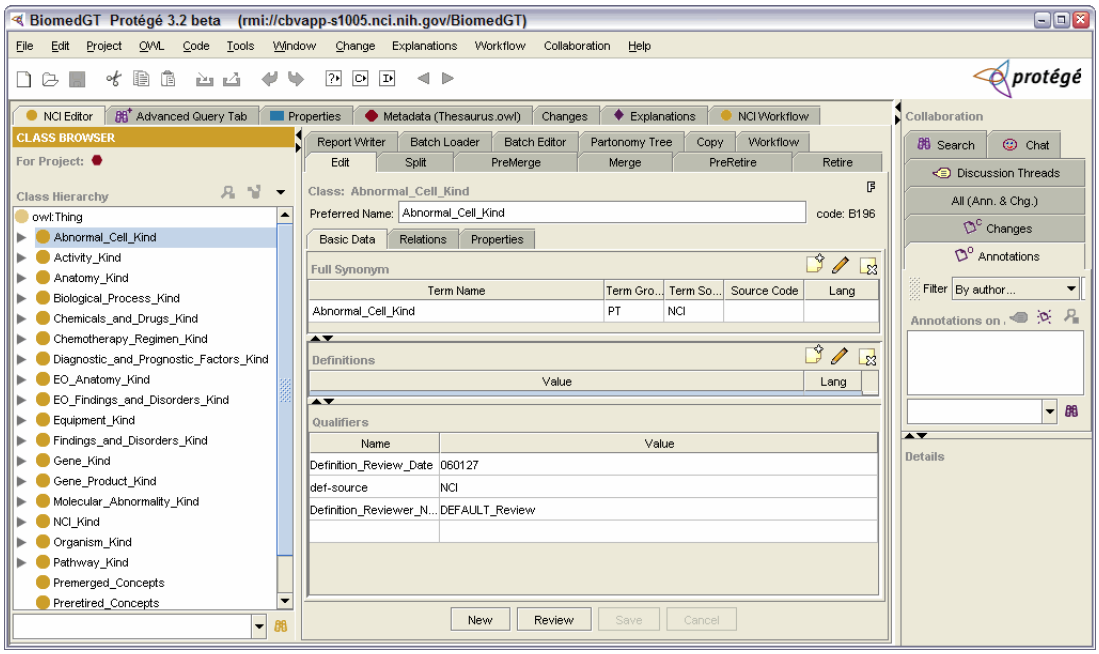


Figure 3.5 Main NCI Protégé window

Using a Local Thesaurus Subset

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

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

Once you set up the tabs and the tab order (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 NCI Protégé. You can select it whenever you want to run the application locally.

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

Downloading the Subset File

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

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

Installing the Subset File

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

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 your 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, then copy the file there. This will keep your project files separate from Protégé application files.

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

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

Opening the Local Subset File in Protégé

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

To open NCI 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-1.2.x`

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

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

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

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

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

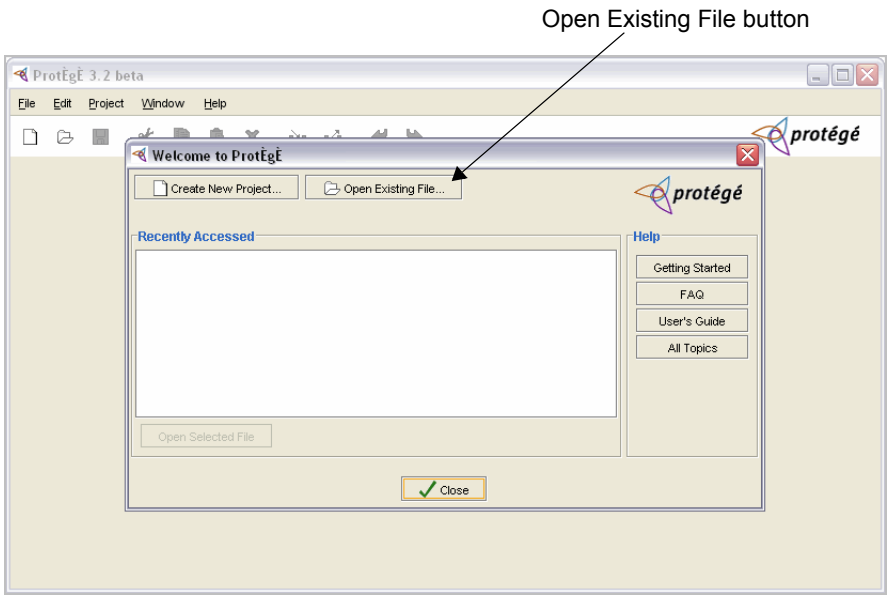


Figure 3.6 Welcome to Protégé window

4. In the **Open Project** window, select **File** (*Figure 3.7* on page 34).
5. Use the **Look In** list to browse for the file.

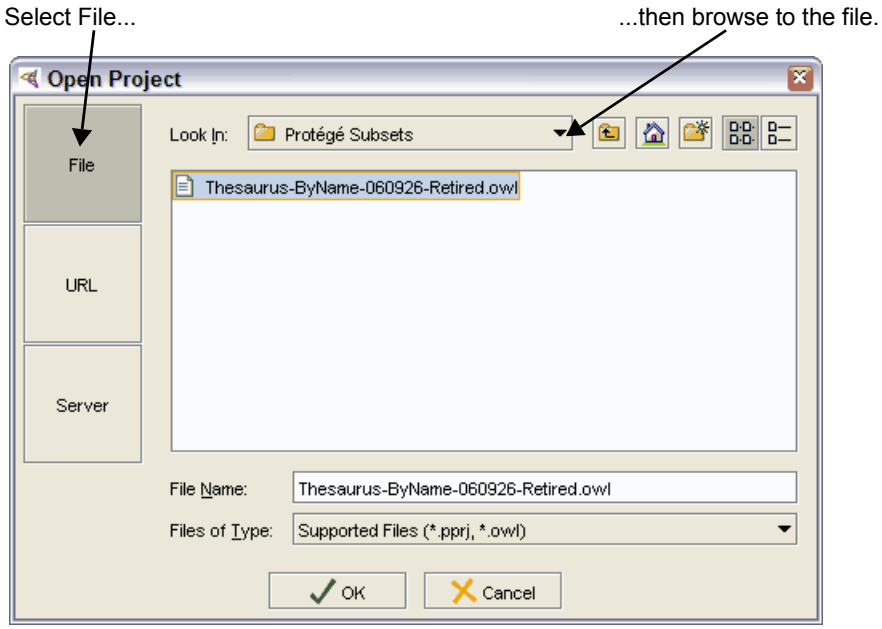


Figure 3.7 Open Project window - File button

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

NCI Protégé opens, but the NCI-specific tabs are not yet visible.

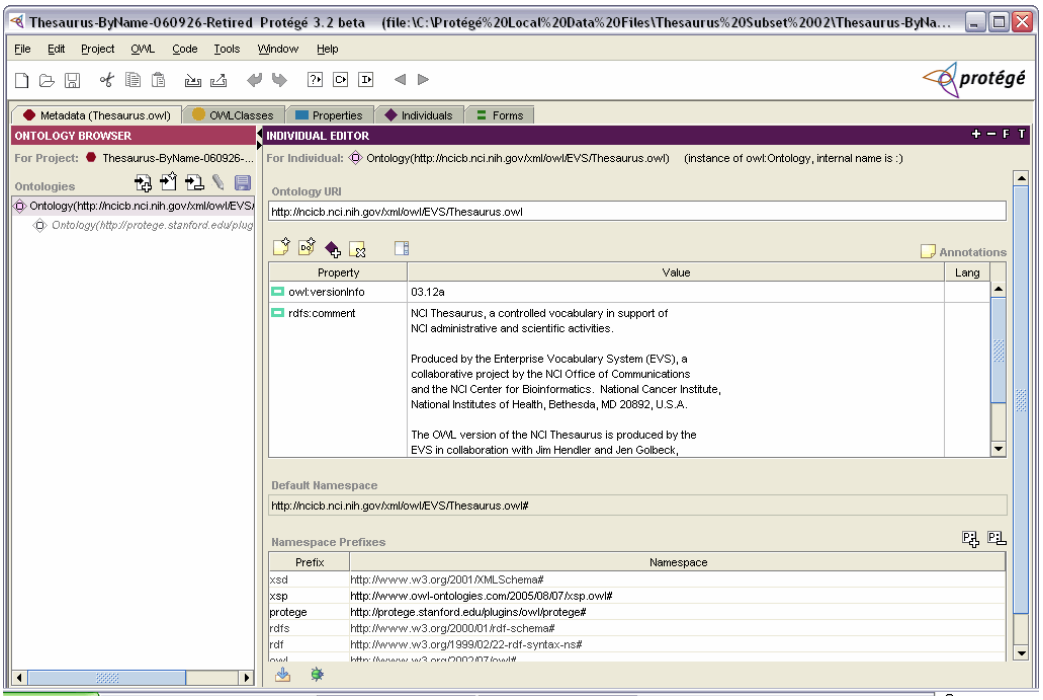


Figure 3.8 Protégé standard interface with no visible NCI-specific tabs

The next section explains how to configure the workspace to mirror the server configuration.

Setting Up a Local Project

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

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

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

   The **Configure File** window opens.

2. Ensure that the **Tab Widgets** tab is selected.
3. In the **Visible** column, uncheck the boxes preceding each checked tab widget.
4. Check the box to the left of the **NCIEditTab** widget.
5. Repeatedly click the **Move Up** button to move the tab to the top of the list.
6. Repeat steps *4.* and *5.* for each of the following tab widgets, moving the widget to the specified position:
   - **AdvancedQueryPlugin** (second position)
   - **OWLPropertiesTab** (third position)
   - **OWLMetadataTab** (fourth position)
   - **ChangesTab** (fifth position)
   - **ExplanationTab** (sixth position)
   - **Workflow Tab** (seventh position)

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

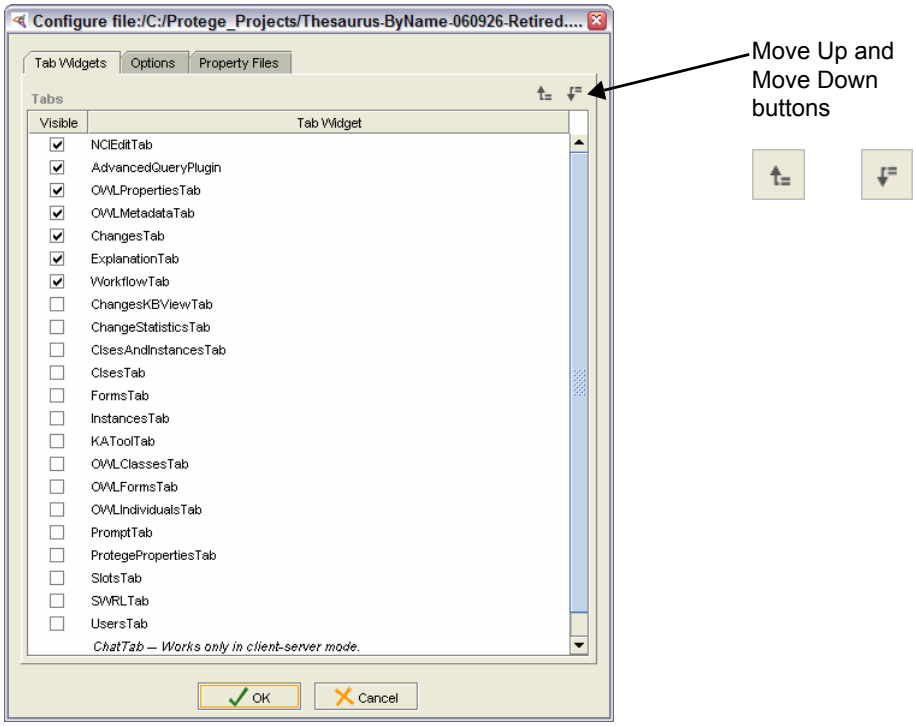


Figure 3.9 Configure File window with ordered tab widgets

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

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

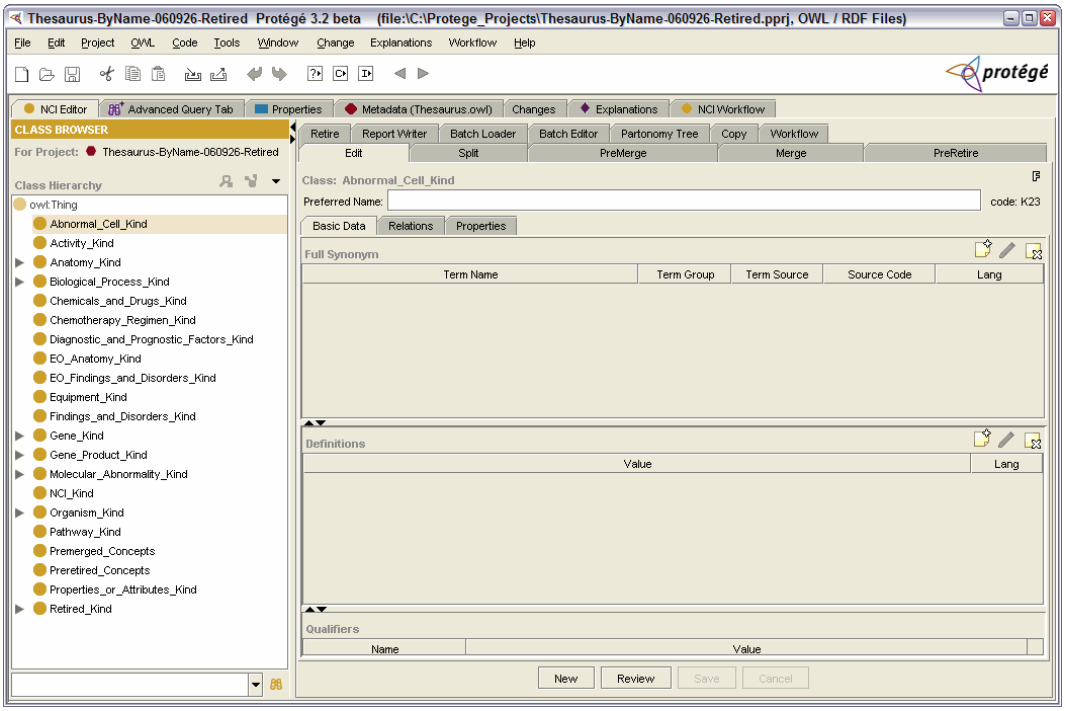


Figure 3.10 New local configuration

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

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

From Here...

- *About the NCI Protégé Work Environment* on page 39
- *Searching* on page 53

CHAPTER

4

ABOUT THE NCI PROTÉGÉ WORK ENVIRONMENT

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

Topics in this Chapter

- *Overview of the Protégé Application* on this page
- *About the Main Protégé Window* on page 40
- *About the NCI Editor Tab* on page 42
- *Working with the Console* on page 46

Overview of the Protégé Application

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.

About NCI Protégé

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

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

About the Main Protégé Window

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

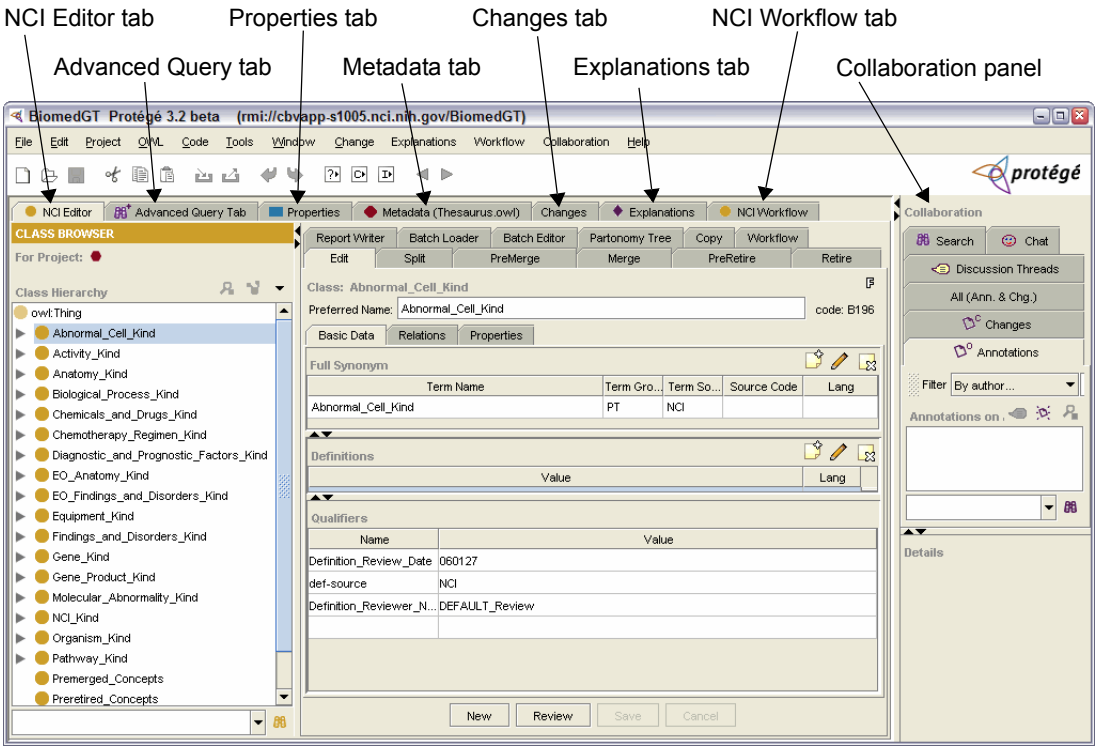


Figure 4.1 Main NCI Protégé window

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

Feature	***Description***
NCI Editor tab	Provides access to a number of subtabs that represent various editing tasks. These include **Edit**, **Split**, **PreMerge**, **Merge**, **PreRetire**, **Retire**, **Report Writer**, **Batch Loader**, **Batch Editor**, **Partonomy Tree**, **Copy**, and **Workflow**. For more information, see *About the NCI Editor Tab* on page 42.
Advanced Query tab	Used for building both simple and complex queries. From this tab, you can link to details for a selected search result. For more detailed information about searching and building advanced queries, see *Building an Advanced Query* on page 57.
Properties tab	Used to look up the definition of any property in the database. Only workflow managers can edit the properties that are available on this tab.
Metadata tab	Used to examine metadata for the entire vocabulary. This includes such information as the creator, the version, and referenced external vocabularies. Only workflow managers can edit the metadata that is available on this tab.

Table 4.1 Description of main window features

Feature	*Description*
Changes tab	Supports the **Prompt** plug-in by analyzing changes between baselines of the main ontology. Though this tab is used mostly by workflow managers, you can use it to trace a series of edit actions by examining the output that is shown in the top panel. To examine changes, you would need to manually step through them.
Explanations tab	Explains why a class was not properly classified by the reasoner. If you generate an inconsistent class, you can use this tab to find out why.
NCI Workflow tab	Used by both workflow managers and editors (modelers) to manage work assignments. Managers use this tab to import packages (proposals) from the BiomedGT wiki and assign workflow items to modelers. Modelers use the tab to view and manage their assignments. For more information, see *Chapter 11, Workflow Procedures for NCI Editors,* on page 165.
Collaboration panel	Supports collaborative ontology editing. Using this panel, you can • Annotate classes, properties, and instances • Annotate changes such as creating or renaming classes • Participate in discussion threads • Initiate proposals and voting • Search and filter annotations using various criteria • Chat with other users who are connected to the NCI Protégé server The Collaboration panel remains visible regardless of which tab you select. To hide this panel, click the **Collapse** button on the top left edge of the panel. To show it again, click the **Expand** button. **Note:** This panel is still under development. For more information about the intended usage and features, visit the following Stanford Web site: http://protege.stanford.edu/doc/collab-protege/

Table 4.1 Description of main window features (Continued)

About the NCI Editor Tab

When you first log in to Protégé, the **NCI Editor** tab is the prominent tab. *Figure 4.2* shows the tab layout.

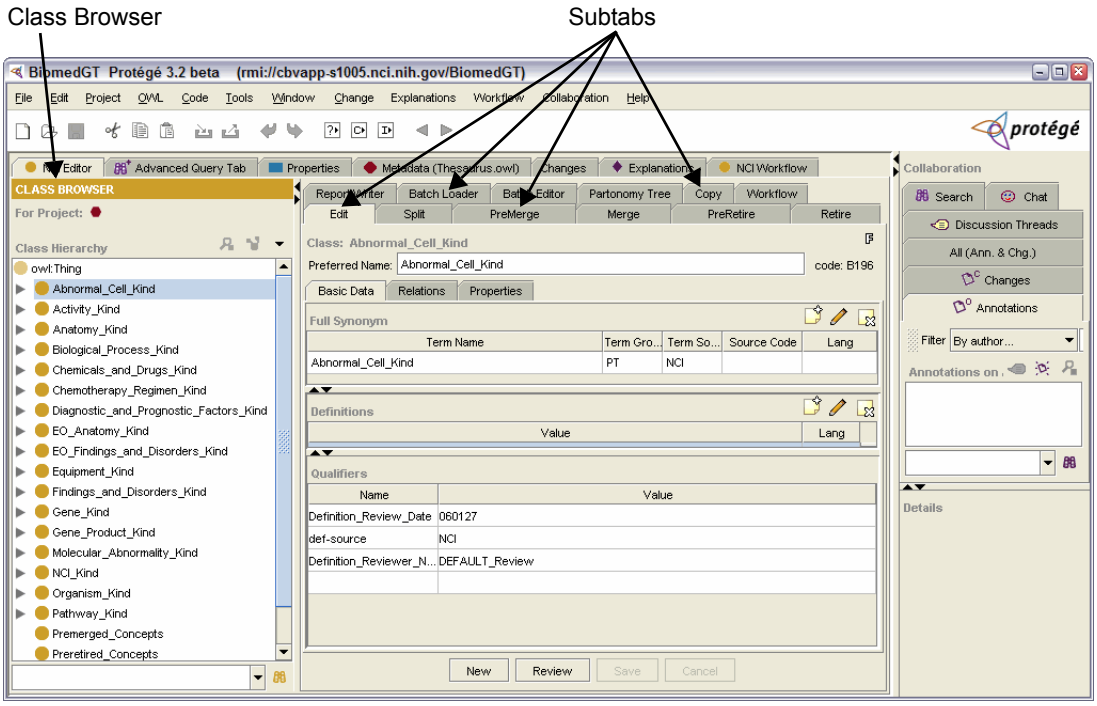


Figure 4.2 NCI Editor tab with Class Browser and subtabs

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

Feature	***Description***
Class Browser	Shows classes and subclasses in a tree view. You can select a class or subclass and view its details using the **Edit** subtab on the right. Below the tree, the Class Browser also includes a **Search** field for entering search terms and a button for executing the search. The results appear in a standalone **Advanced Query** window. For more detailed information, see *Class Browser* on page 43.
Subtabs	Enable you to perform various editing tasks. This guide discusses each subtab in the context of the task that it represents. For a description of each subtab and cross-references to more detailed information, see *Table 4.4* on page 44.

Table 4.2 NCI Editor tab: main features

Class Browser

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

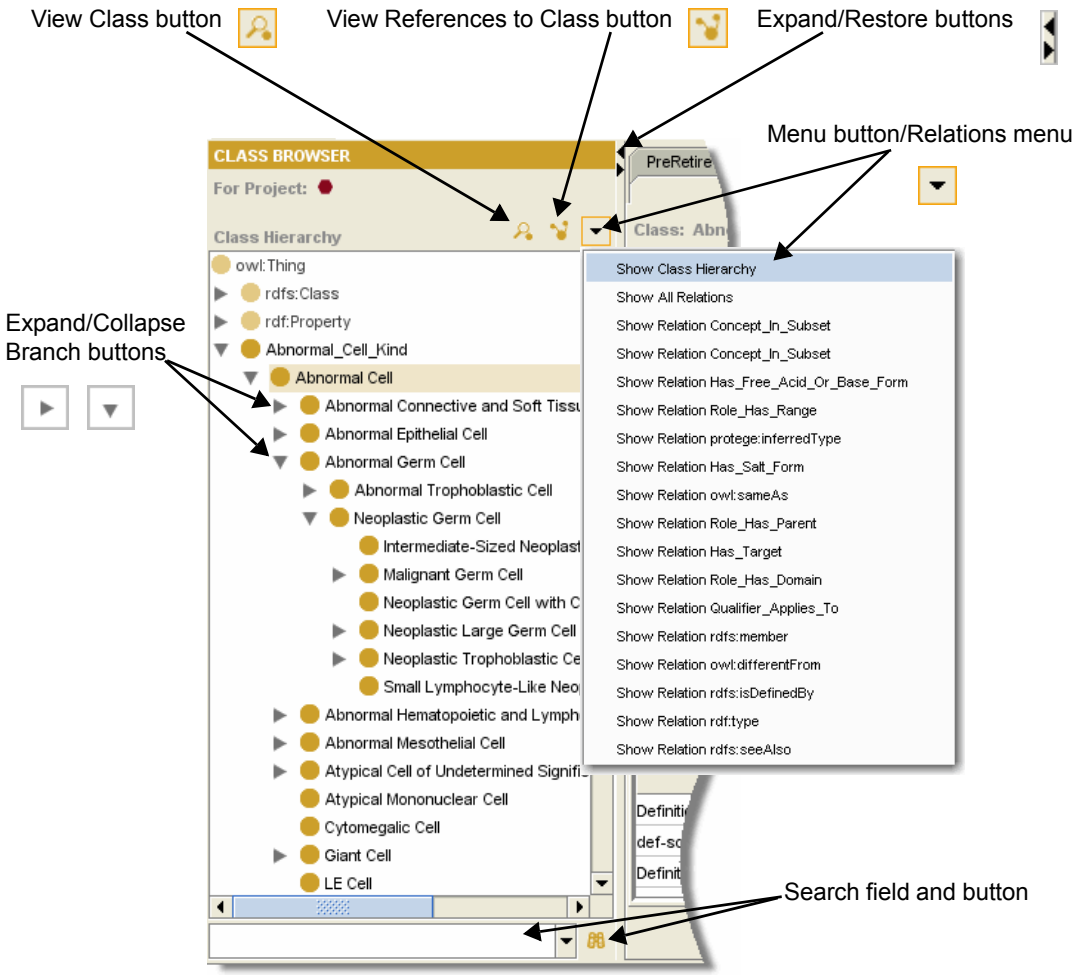


Figure 4.3 Class Browser

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

Feature	***Description***
View Class button	Opens a read-only window showing basic data, relations, and properties for the selected class.
View References to Class button	Opens a read-only window showing references to the selected class.
Expand/Restore buttons	• **Expand** button: Expands the browser to fill the entire window. • **Restore** button: Restores the browser to the left panel.

Table 4.3 Class Browser features

Feature	*Description*
Menu button/**Relations** menu	Shows the **Relations** menu, from which you can select a number of data views. To restore the view to show all classes, select the **Show Class Hierarchy** command.
Expand/Collapse Branch buttons	Shows/hides subclasses of a selected class.
Search field and button	Enable you to type a search string and execute a search.

Table 4.3 Class Browser features

NCI Editor Subtabs

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

Subtab	*Description*	*More Information*
Edit	Displays annotation and relation data for a selected class. You can use the **Basic**, **Relations**, and **Properties** subtab to view and edit properties, restrictions, and associations.	See *Examining Class Information* on page 84.
Split	Creates a new class from an existing class.	See *Splitting a Class* on page 123.
PreMerge	Enables you to select and flag two classes for a merge. Although you can apply a pre-merge flag to classes, only workflow managers can perform an actual merge.	See *Merging Classes* on page 127.
Merge	(Workflow managers only) Enables a workflow manager to merge two classes that have been flagged using the pre-merge action.	See *Merging Classes* on page 127.
PreRetire	Enables you to flag a class for retirement. Although you can apply a pre-retire flag to a class, only workflow managers can retire a class.	See *Chapter 9, Pre-Retiring and Retiring Classes,* on page 137.
Retire	(Workflow managers only) Enables a workflow manager to retire a class that has been flagged by a pre-retire action.	See *Chapter 9, Pre-Retiring and Retiring Classes,* on page 137.
Report Writer	Generates reports for selected classes. You can save reports as external files and share them with colleagues.	See *Generating Reports for Selected Classes* on page 147.

Table 4.4 Descriptions of NCI Editor subtabs

Subtab	*Description*	*More Information*
Batch Loader	Enables you to load a batch of classes from an external file, save them to the Protégé database, and edit them.	See *Loading a Batch of Classes for Editing* on page 154.
Batch Editor	Enables you to load a batch of classes that have been edited in an external application and save them to the Protégé database.	See *Loading a Batch of Edited Classes* on page 159.
Partonomy Tree	Shows classes that are connected by *part_of* relations. This tab enables you to select a root class and display it as a partonomy tree.	See *Viewing Classes Connected by part_of Relations* on page 97.
Copy	Clones a class from an existing class; also facilitates simultaneous editing of two classes.	See *Cloning a Class* on page 133.
Workflow	Used to examine concepts related to incoming proposals from the BiomedGT wiki.	See *Chapter 11, Workflow Procedures for NCI Editors,* on page 165.

Table 4.4 Descriptions of NCI Editor subtabs

Working with the Console

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

Figure 4.4 shows the Console as it appears after NCI Protégé first opens.

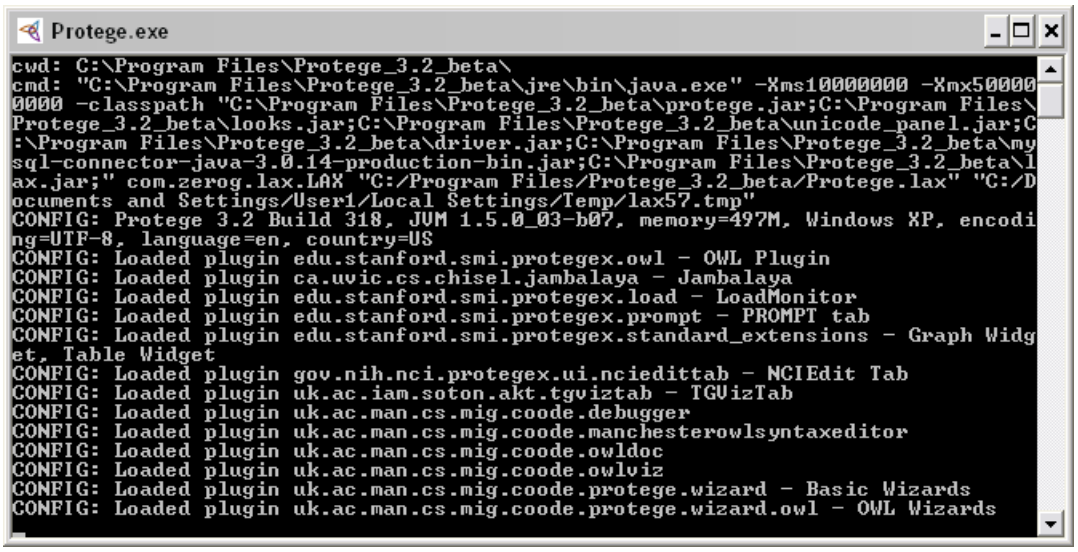


Figure 4.4 Console window

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

Topics in This Section

- *Setting the Console Display Capacity* on page 47

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

- *Capturing Information for Troubleshooting* on page 48
- *Submitting a Bug Report and File Attachment to GForge* on page 49

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 (*Figure 4.5*).

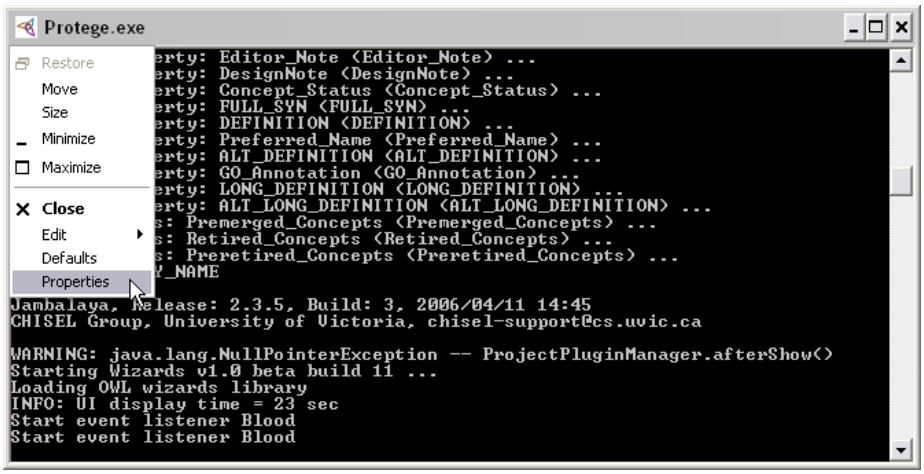


Figure 4.5 Console window menu - Properties command

The **Properties** window opens (*Figure 4.6*).

3. Click the **Layout** tab.
4. In the **Screen Buffer Size** section, type `9999` in the **Height** field (*Figure 4.6*).

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

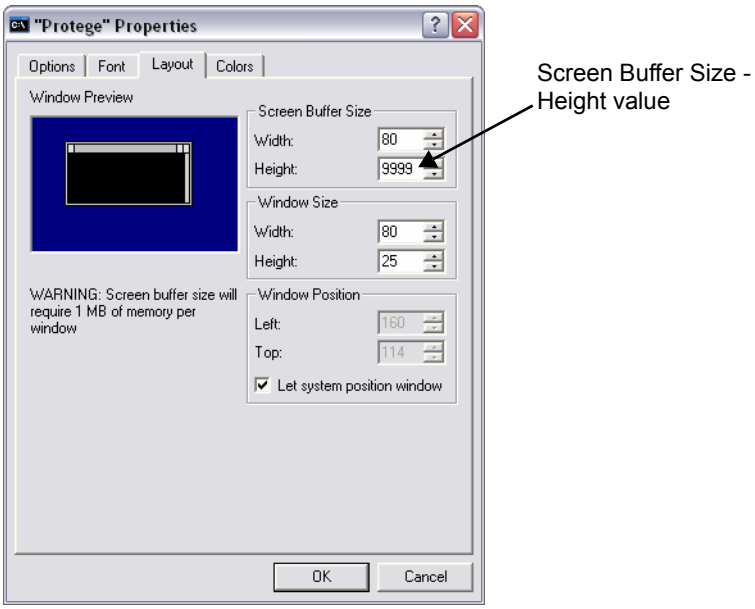


Figure 4.6 Console Properties window

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

Capturing Information for Troubleshooting

Prerequisite task: *Setting the Console Display Capacity* on page 47

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 the GForge Tracker.

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
   - right-clicking the blank area of the window title bar.
2. Select **Edit** > **Select All** to select all of the text in the window (*Figure 4.7*).

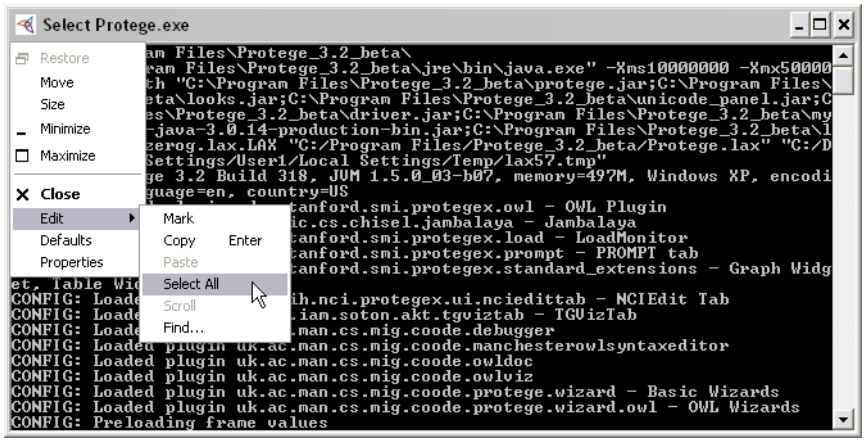


Figure 4.7 Command Line window menu - Select All command

3. Access the Console menu again, then select **Edit** > **Copy**.
4. Open a text editor such as Windows Notepad.
5. Paste the copied text into a blank text file.
6. Save the file with a name such as `console-copy_<date>.txt`, where *<date>* is today's date in MM-DD-YYYY format (for example, `console-copy_02-12-08`.txt).

Submitting a Bug Report and File Attachment to GForge

While working with Protégé, you may occasionally find a problem that you would like to report to the development team. You can use the **Tracker** tab on GForge to report a bug.

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

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

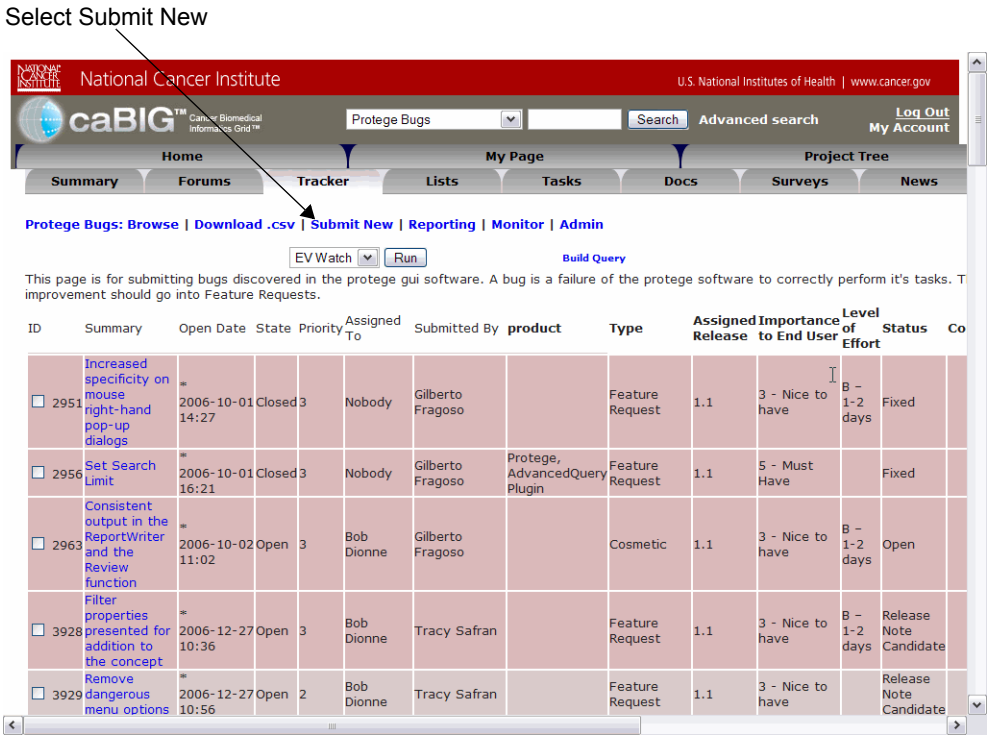


Figure 4.8 Tracker page - Submit New link

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

Field	***Description***
Level of Difficulty	Leave as is. The developer assigned to fix the bug sets the value.
Product	Select the name of the product in which the defect was found.
Type	Select a description for the nature of the problem.

Table 4.5 Bug Tracker fields

Field	Description
Assigned Release	Select the appropriate release number from the list.
Importance to End User	Select a priority level.
Level of Effort	Do not use this field. It will eventually be removed from the form.
Status	Leave as is. The Development team assigns the status indicator.
Component	Select the application component to which the problem applies.
Level of Effort - Low (in Days)	Leave as is. The Development team uses it to show the minimum estimated level of effort to fix the defect (code and unit test) in working days (assumes 100% resources working eight-hour days).
Level of Effort - High (in Days)	Leave as is. The Development team uses it to show the maximum estimated level of effort to fix the defect (code and unit test) in working days (assumes 100% resources working eight-hour days).
Previous Build Tested	Leave as is. The QA team uses it to show the version number of the most recent application version used to test the problem. Testers change this number whenever they verify a developer's fix. If the status of a problem is *Fixed*, this field tells you the exact version that contains the fix.
Version Found	Enter the version number of the application that always reproduces the problem. The QA team uses this field to track the defect, so it must always show the correct version.
Assigned To	Leave as is. The Development team uses it to assign the request to a team member.
Priority	Leave as is. The Development uses it to assign a priority level between 1 (lowest) and 5 (highest).
Summary	Add a summary title for the problem.
Detailed Description (with Notepad icon)	Add a detailed description of the problem, including the task were you performing when the problem happened. If you were able to capture the text of any messages that appeared, include them here.
Check to Upload & Attach File	1. Check the box if you are including a file. 2. Click **Browse**, then find the file that you created in *Copying the Console Window Contents* on page 48. 3. Upload the file.
File Description	Enter a brief description of the file that you included.

Table 4.5 Bug Tracker fields (Continued)

9. Click **Submit**.

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

From Here...

- *Searching* on page 53
- *Working with Classes: The Basics* on page 81

CHAPTER 5

SEARCHING

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

Topics in this Chapter

- *Simple vs. Advanced Search* on this page
- *Performing a Simple Search* on page 54
- *Building an Advanced Query* on page 57
- *Configuring the Advanced Query Tab* on page 75

Simple vs. Advanced Search

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

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

Figure 5.1 Simple search field and button

Note: For more information, see *Performing a Simple Search* on page 54.

- 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 an Advanced Query* on page 57.

Performing a Simple Search

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

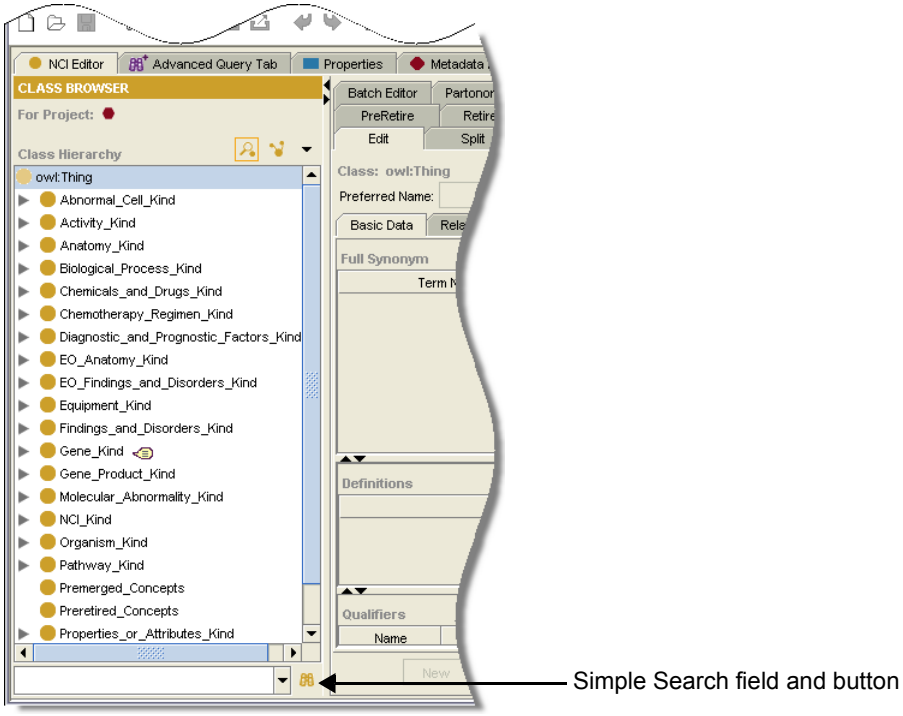


Figure 5.2 NCI Editor tab with simple search field and button

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

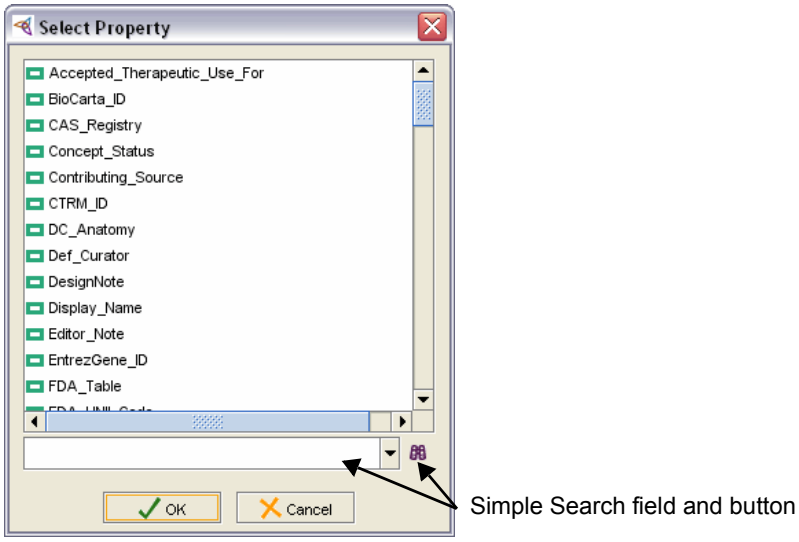


Figure 5.3 Select Property window with simple search field and button

Using the Simple Search Field and Button

To perform a simple search, follow these steps:

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

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

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

Figure 5.4 Simple search field and Search for Class button

The **Advanced search** window opens (*Figure 5.5*).

Note: The layout of this window is identical to the layout of the **Advanced Query** tab, discussed in *Building an Advanced Query* on page 57. For simple searches, use the features in the right side of the window to select, search for, and display results.

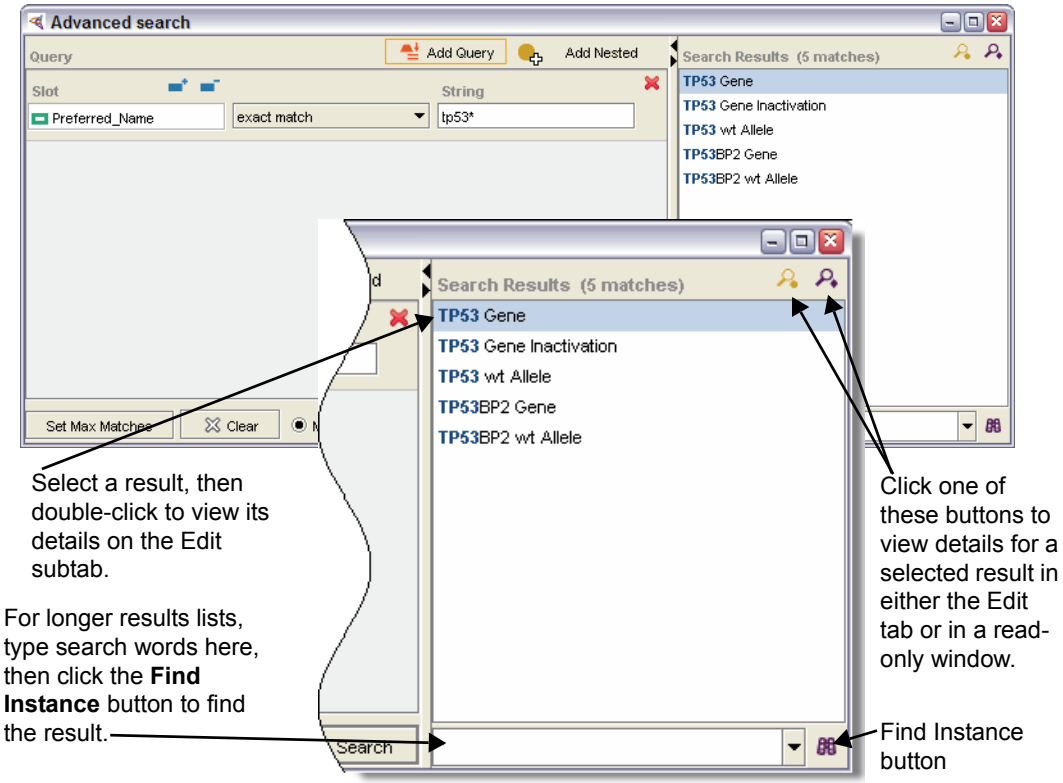


Figure 5.5 Results for simple search

About the Search Results Display

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

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

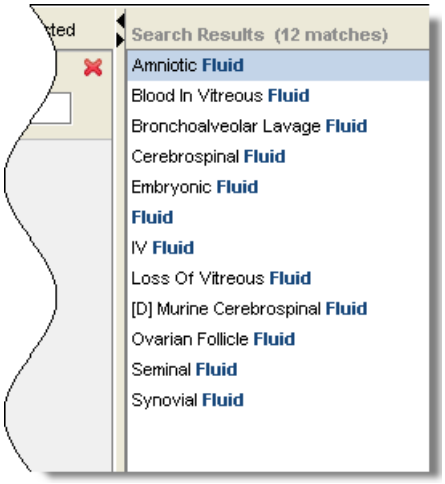


Figure 5.6 Results: Preferred_Name ends with fluid

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

Selecting and Reviewing Search Results

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

1. (Optional) To search through a long results list, follow these steps:
   a. Type a search word or phrase in the **Search** field below the list (*Figure 5.5* on page 55).
   b. Click the **Find Instance** button .
2. View details for a result using any of the following methods:
   - Double-click the selected result to view its basic data, relations, and properties on the **NCI Editor** tab > **Edit** subtab.
   - Select the result, then click the **Edit Class** button to view the basic data, relations, and properties for the selected result on the **NCI Editor** tab > **Edit** subtab.
   - Select the result, then click the **View Instance** button to view the basic data, relations, and properties for the selected result in a standalone, read-only window. The window shows in the foreground.

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

Building an Advanced Query

An advanced query gives you more control over the scope and complexity of searches. Using an advanced query, you can

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

The **Advanced Query** tab (*Figure 5.7*), enables you to build complex queries. *Figure 5.7* shows the layout of this tab. For an explanation of the main features, see *Table 5.1* (page 58).

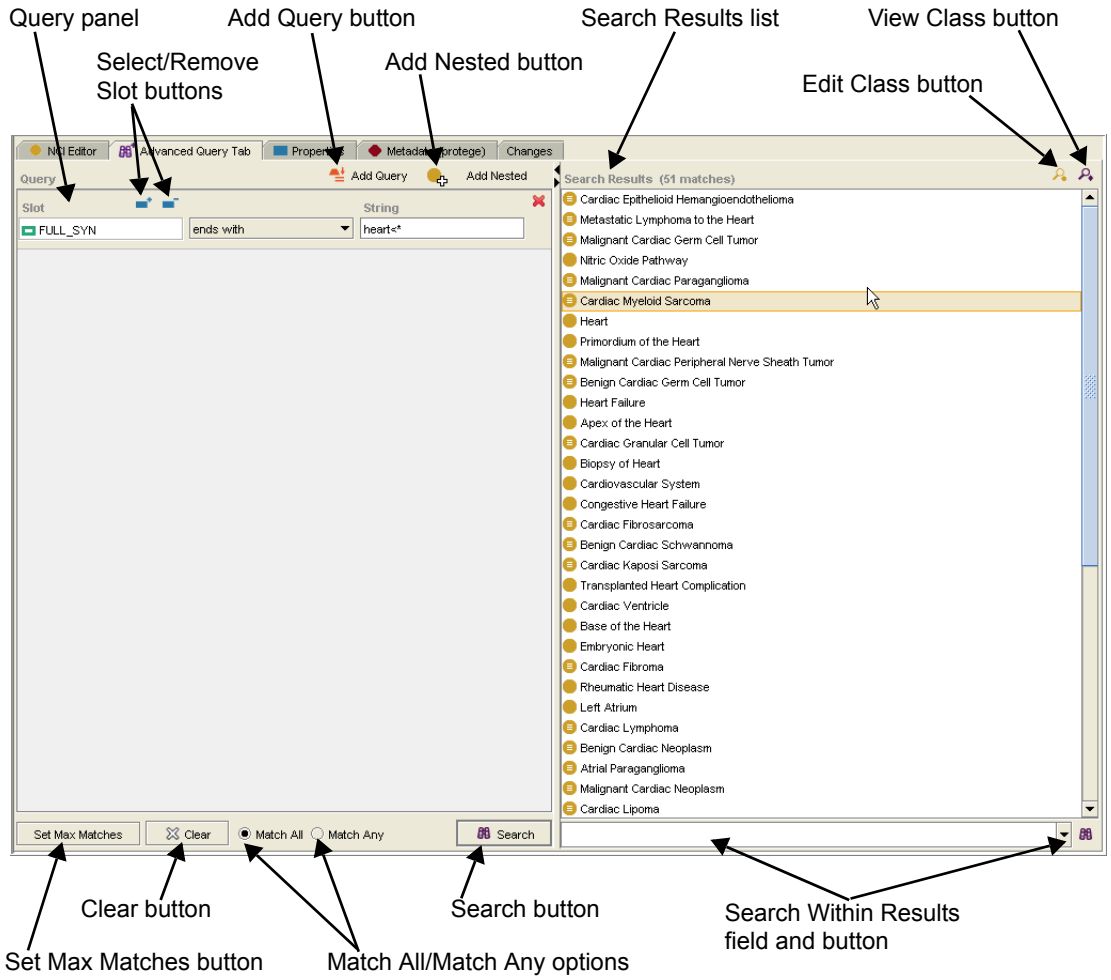


Figure 5.7 Advanced Query tab

Table 5.1 lists and describes the main of the **Advanced Query** tab.

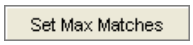
Feature	***Description***
Query panel	Shows a full search expression: slot, parameters (*e.g.*, *contains*, *starts with*) and a typed string.
Select Slot button	Opens a window in which you can select a new slot (property) to replace the current slot.
Remove Slot button	Clears the **Slot** field, clears the parameters list, and removes the **String** field. To restore these features and add new query information, click the **Select Slot** button.
Add Query button	Adds a new query, including a **Slot** field (with *Preferred_Name* displayed), a parameters list, and an empty **String** field. Use these tools to build your own expressions.
Add Nested button	Enables you to build a query for finding classes with specific restrictions or associations.
Search Results list	Lists the returned results from a search and enables you to search within the list. See *Figure 5.7*.
Edit Class button	Shows the basic data, relations, and properties of a selected search result in the **NCI Editor** tab > **Edit** subtab.
View Class button	Opens a read-only window in which you can view basic data, relations, and properties for a selected search result.
Set Max Matches button 	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: *Preferred_Name contains <blank>*. 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. **Match All** (the default) performs an AND search, and **Match Any** performs an OR search.
Search button	Executes a search.
Search Within Results field and button	Enable you to search for an item from the returned results list. Type a term in the field and click the button on the right to execute the search.

Table 5.1 Advanced Query tab features

Advanced Query Syntax

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

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

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

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

```
contains >heart<

starts with *>heart

ends with heart<*

exact match *>heart<*
```

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

Tips for Using the Query-Building Interface

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

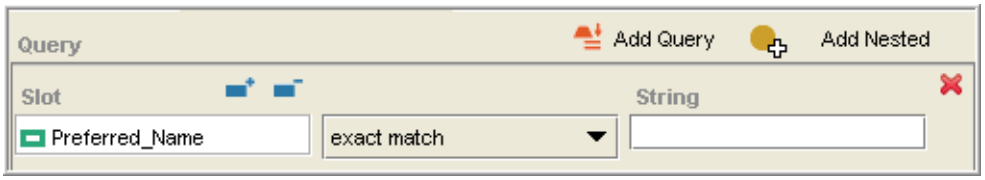


Figure 5.8 Default query

- 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.

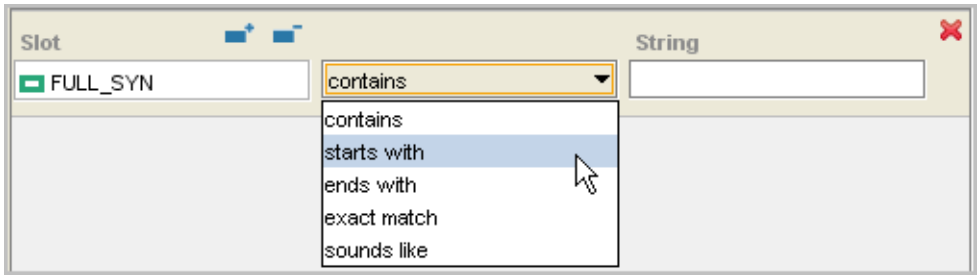


Figure 5.9 Parameter list

- Type a search string in the **String** field.
- Add queries using the **Add Query** button

.
- Use the **Add Nested** button

to add a nested query when searching for restrictions or associations. For more information, see *Building a Restriction-Based Query* on page 67.
- To remove a single query, click the **Remove Query** button . If you have built a single query and then click this button, Protégé restores the query to the default query (*Preferred_Name* with exact matching).
- Click the **Clear** button

to remove all queries. After clearing the queries, Protégé displays the default query shown in *Figure 5.8*.
- The default search scope is **Match All**, which uses a Boolean AND. To test the results of your queries, try changing the scope option to **Match Any**, which uses a Boolean OR

.

From Here...

The rest of this section details procedures for building queries of varying complexity:

- *Building a Simple Property Query: Preferred Name* on page 61
- *Building a Complex Property Query: FULL_SYN* on page 62
- *Building a Combined Property Query: Preferred_Name and FULL_SYN* on page 64
- *Building a Restriction-Based Query* on page 67
- *Building an Association-Based Query* on page 70
- *Exporting Your Search Results* on page 72

Building a Simple Property Query: Preferred Name

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

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

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

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

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

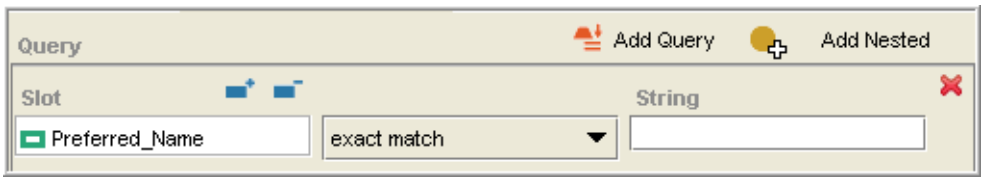


   Figure 5.10 Default query

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

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

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

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

Building a Complex Property Query: FULL_SYN

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

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

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

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

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

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

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

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

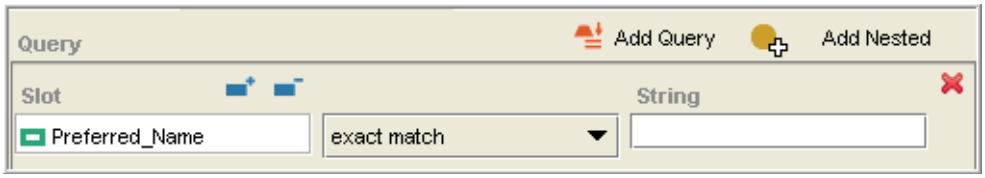


Figure 5.11 Default query

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

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

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

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

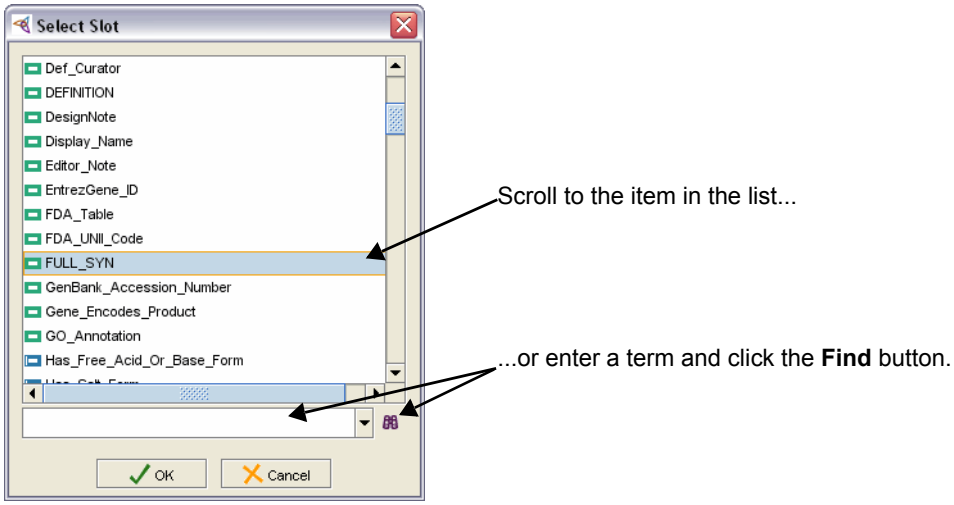


Figure 5.12 Select Slot window

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

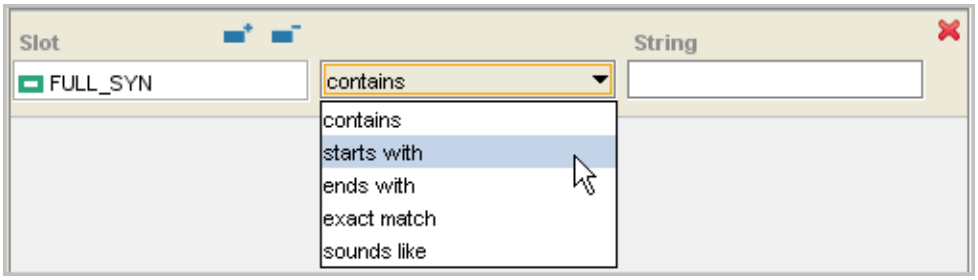


Figure 5.13 Parameter list - starts with

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

   The full query is complete.

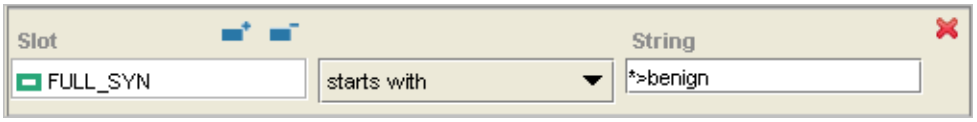


*Figure 5.14 Finished query - FULL_SYN starts with *>benign*

7. Leave the search scope set to **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 **NCI Editor** > **Edit** subtab, which now displays FULL_SYNs, definitions, and qualifiers for the selected class.

Building a Combined Property Query: Preferred_Name and FULL_SYN

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

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

Procedure: To build and execute a combined property query, follow the steps in the next four sections.

*First Query: Preferred_Name starts with bcl*gene*

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

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

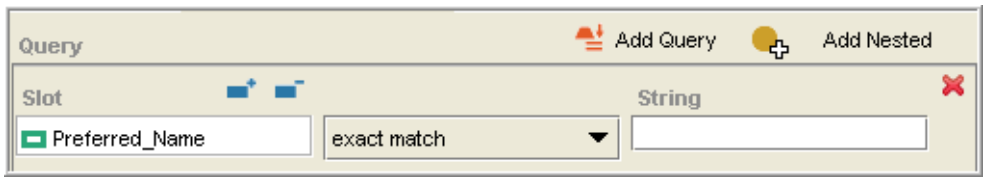


Figure 5.15 Default query

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

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

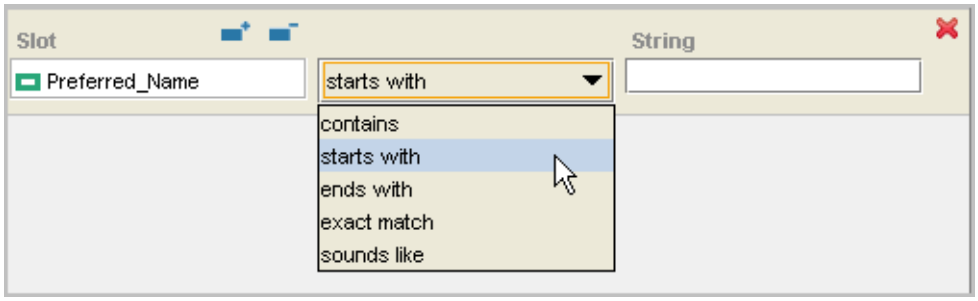


Figure 5.16 Selecting from parameter list

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

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

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

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

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

1. 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**.

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

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

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

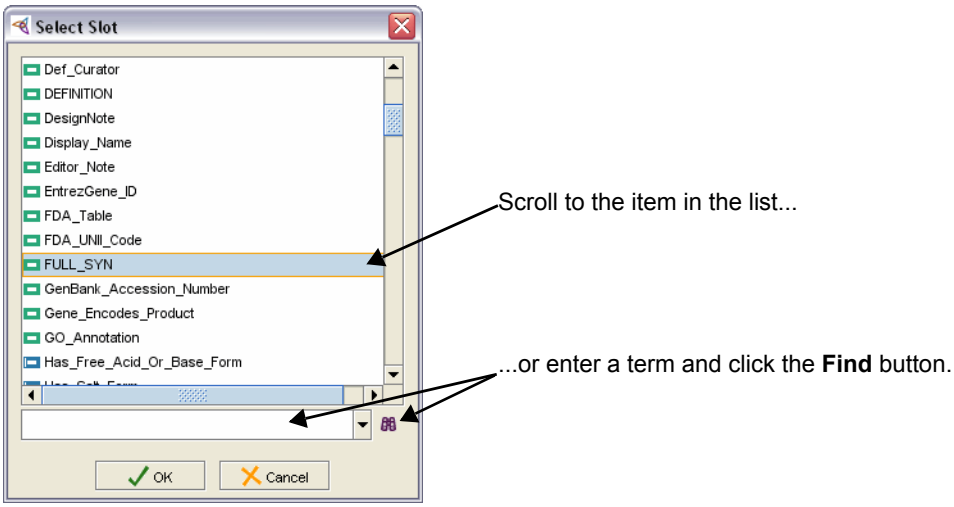


Figure 5.18 Select Slot window

4. Click **OK** to close the window.
5. Change the search parameter to **starts with**.
6. In the **String** field, type `*>bcl*gene` (*Figure 5.19*).

   Since FULL_SYN is a complex property, include a preceding angle bracket.

*Figure 5.19 Completed query - FULL_SYN starts with *>bcl*gene*

*Third Query: Preferred_Name starts with bcl*allele*

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

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

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

   The combined queries should now resemble *Figure 5.20*.

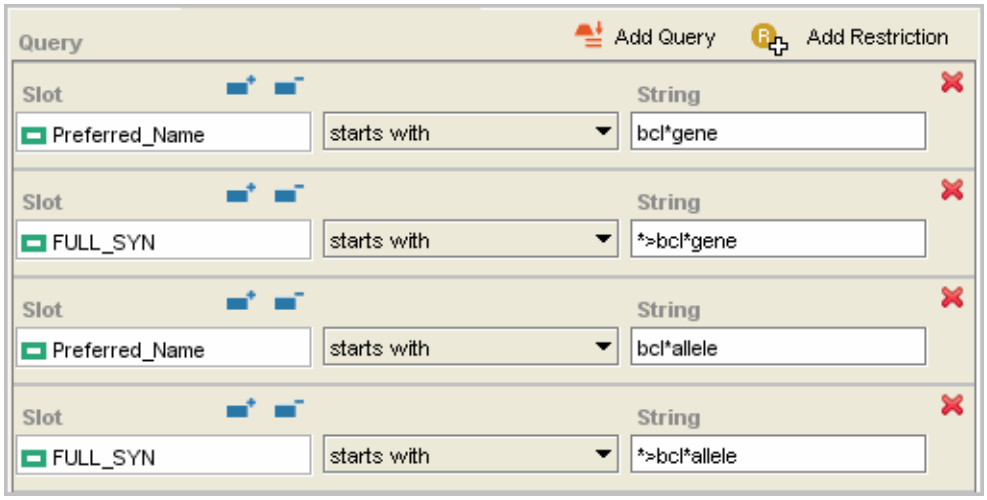


Figure 5.20 Combined queries

5. Change the search scope to **Match Any** (Boolean OR).
6. Click the **Search** button .

   Figure 5.21 (page 66) shows the full query with results.

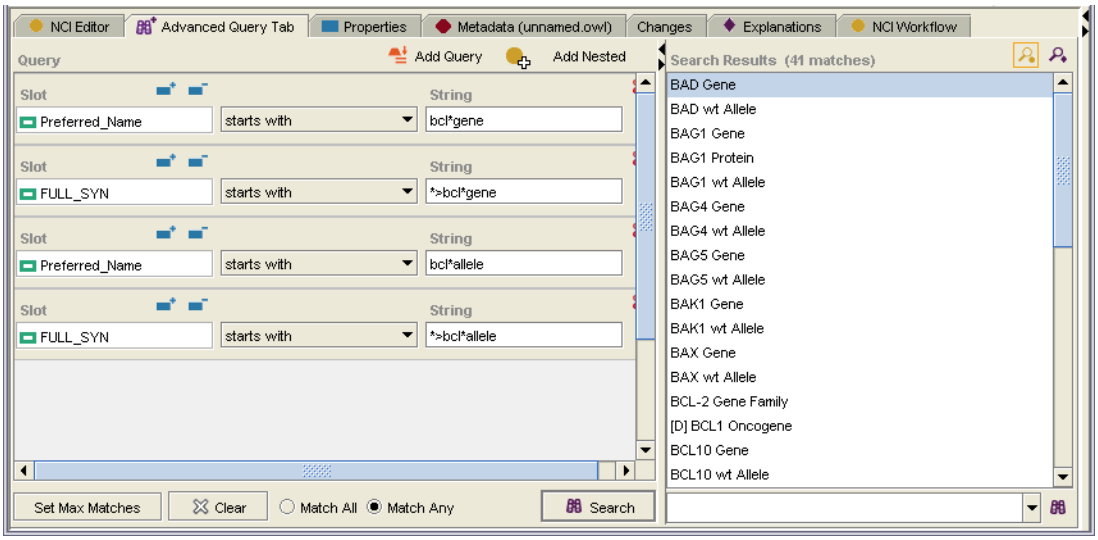


Figure 5.21 Full query with search results

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

On Your Own...

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

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

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

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

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

.

   Protégé places the restriction below the default query. Since you do not need the default query, you can remove it.
3. Delete the default query by clicking the first instance of the **Remove Query** button (*Figure 5.22*).

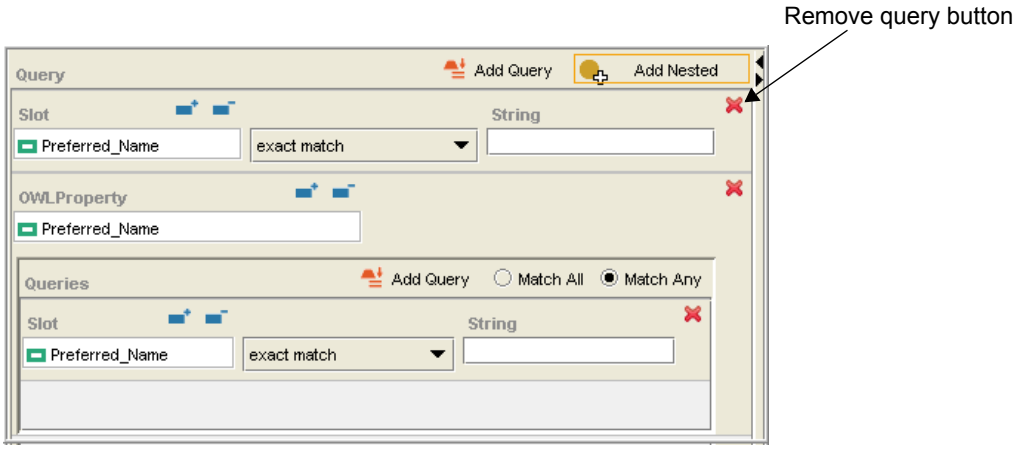


Figure 5.22 Removing the default query

The left side of the **Advanced Query** tab should now resemble *Figure 5.23*.

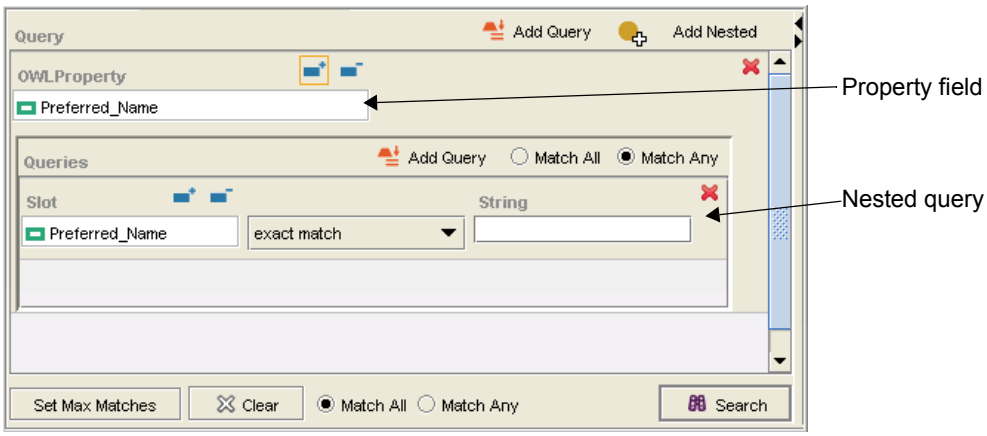


Figure 5.23 Initial setup for restriction-based query

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

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

   The **Select Property** window opens (*Figure 5.24*)

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

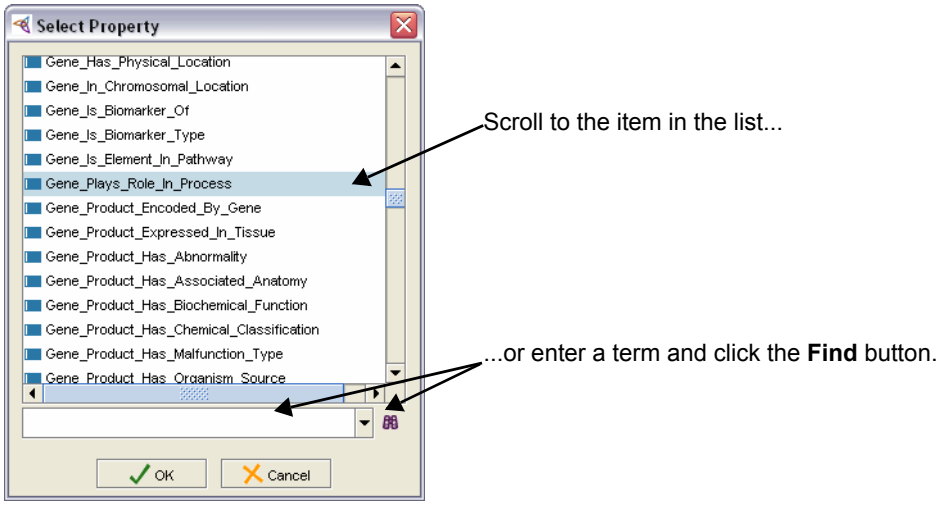


Figure 5.24 Select property window

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

The upper **Slot** field now shows *Gene_Plays_Role_In_Process* (*Figure 5.25*).

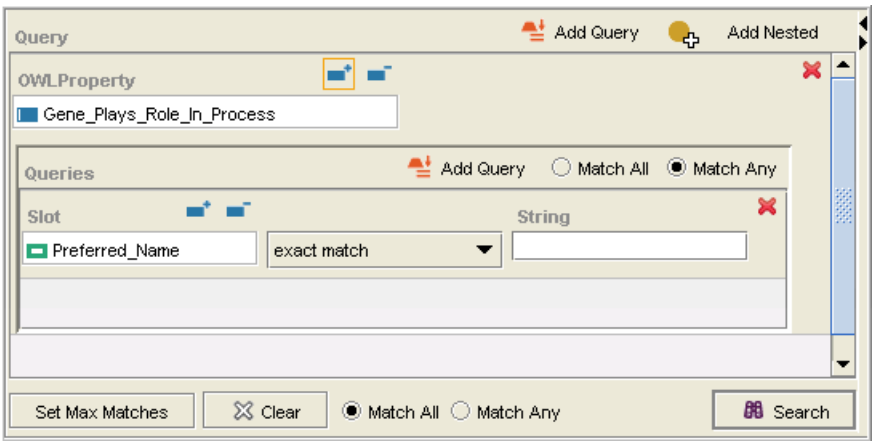


Figure 5.25 Restriction-based query with search property

7. In the nested **Queries** panel, leave the slot set to **Preferred_Name**.
8. Leave the parameter set to **exact match**.
9. In the **String** field, type `oncogenesis` (lowercase or uppercase). The query should now resemble *Figure 5.26*.

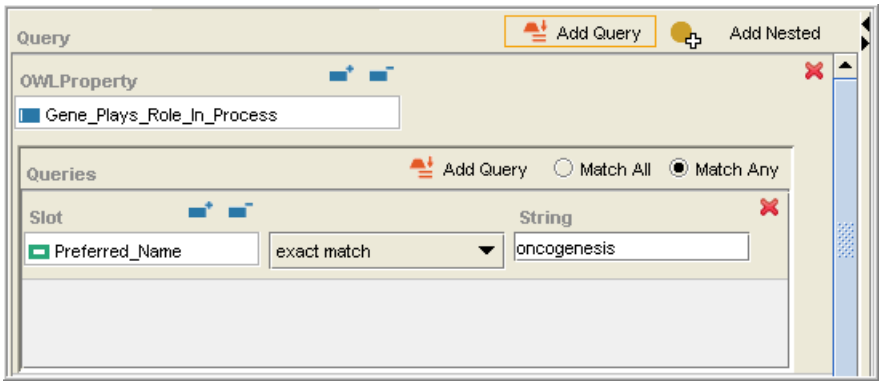


Figure 5.26 Restriction-based query with completed fields

10. Click the **Search** button .

The search results appear on the right panel. (*Figure 5.27*).

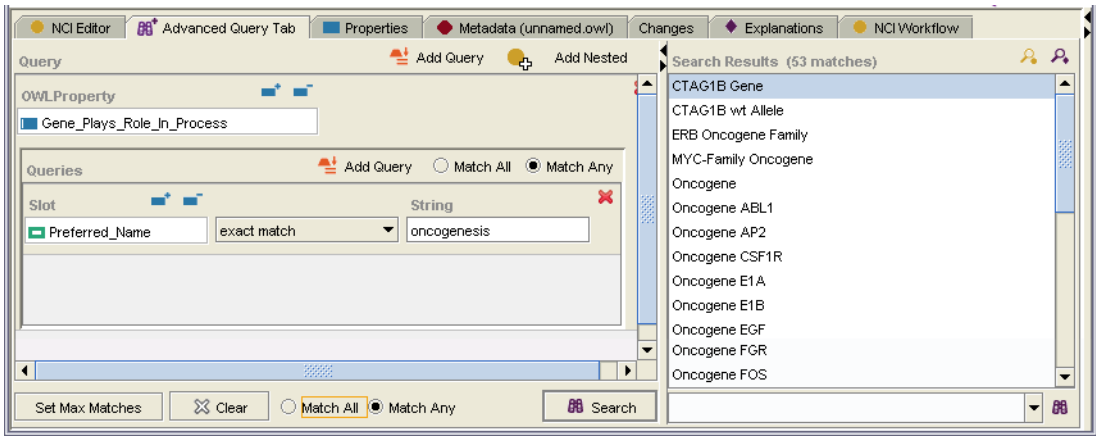


Figure 5.27 Search results for restriction-based query

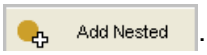
11. To view details for a selected result, double-click the result.

Building an Association-Based Query

This section explains how to search by association.

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

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

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

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

3. Delete the default query by clicking the first instance of the **Remove Query** button (*Figure 5.28*).

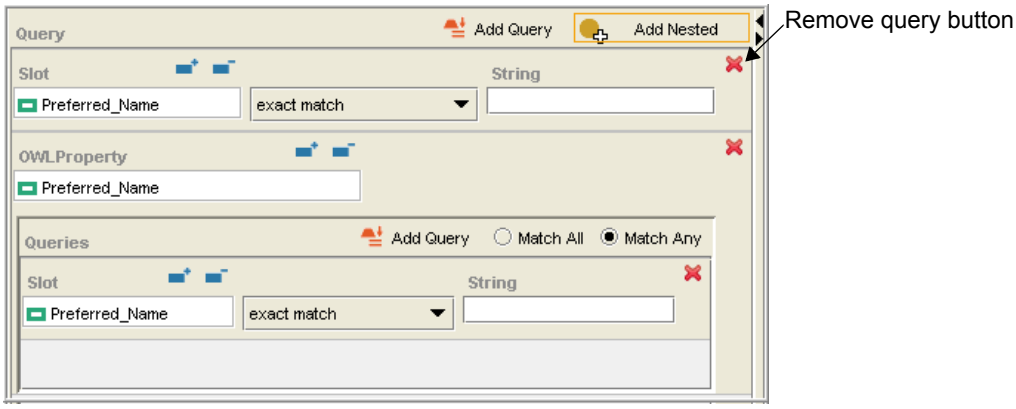


Figure 5.28 Removing the default query

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

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

   The **Select Property** window opens.

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

   The query should now resemble *Figure 5.29*, with the *upper* **Slot** field showing *Concept_In_Subset*, and the lower fields showing the following:

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

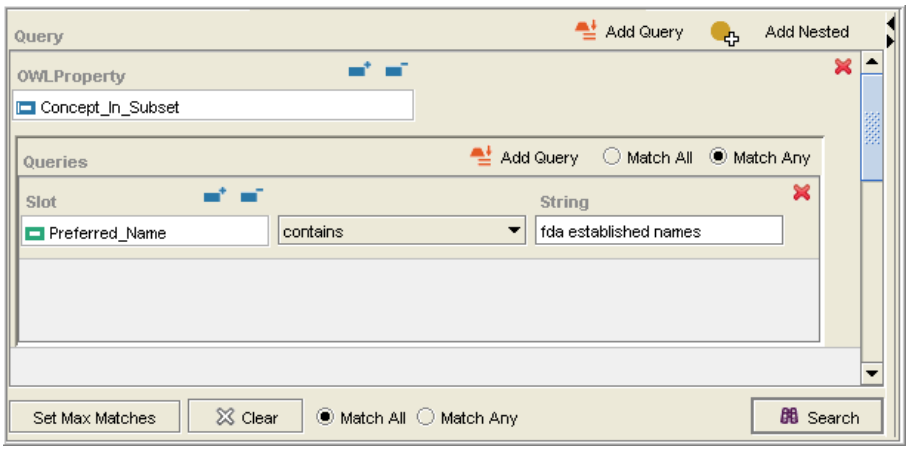


Figure 5.29 Association-based query with search property

10. Click the **Search** button .

    The search results appear on the right panel.

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

Exporting Your Search Results

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

Scenario: You want to find concepts for which the **Preferred_Name** contains the word *gene*.

Procedure: To build and execute the query, follow the steps in the next three sections.

Building the Query

1. Click the **Advanced Query** tab.
2. Set the following parameters:
   - **Slot:** Preferred_Name
   - **Search parameter:** contains
   - **String:** gene
3. Click the **Set Max Matches** button.
4. In the **Input** window, set the max matches count to **50**, then click **OK**.
5. Click the **Search** button.

Starting the Export

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

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

   The **Export configuration** window opens. A default file path and name appears in the **Exported file** field (*Figure 5.30*).

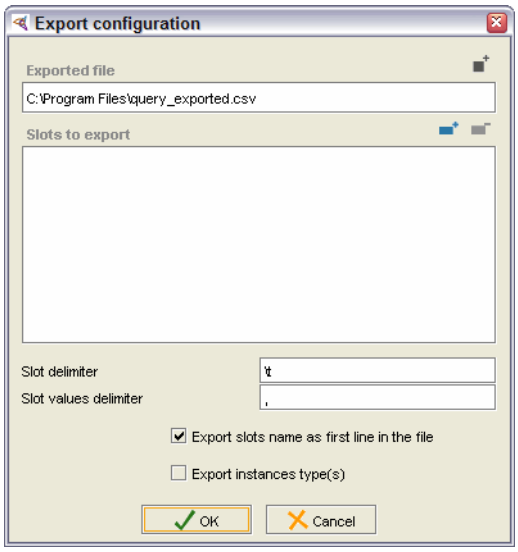


Figure 5.30 Export configuration window

2. (Optional) If you prefer to rename the file and save it to a different directory, follow these steps:
   a. Click the **Browse for File** button .
   b. Navigate to the desired directory.
   c. Change the **Files of Type** list to show **All Files**.
   d. (Optional) Change the file name to your preferred name.
   e. (Required) Change the file extension from `.csv` to `.txt`.

   Note: Testing has shown that a `.txt` file with Tab and Pipe delimiters creates the best output for this purpose.

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

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

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

   This step determines the delimiter used to separate the various properties.
5. In the **Slot values delimiter** field, delete the comma and replace it with a **Pipe** symbol (|).

   On the keyboard, the Pipe symbol shares a key with the Backslash (\) symbol. Press SHIFT + \ to produce the Pipe symbol. This symbol will be used as the delimiter that separates property values.
6. Check the box labeled **Export slots name as first line in the file**.

   This step establishes the column headings in the output.
7. (Optional) Check the box labeled **Export instances type(s)**.

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

The **Export configuration** window should now resemble *Figure 5.31*.

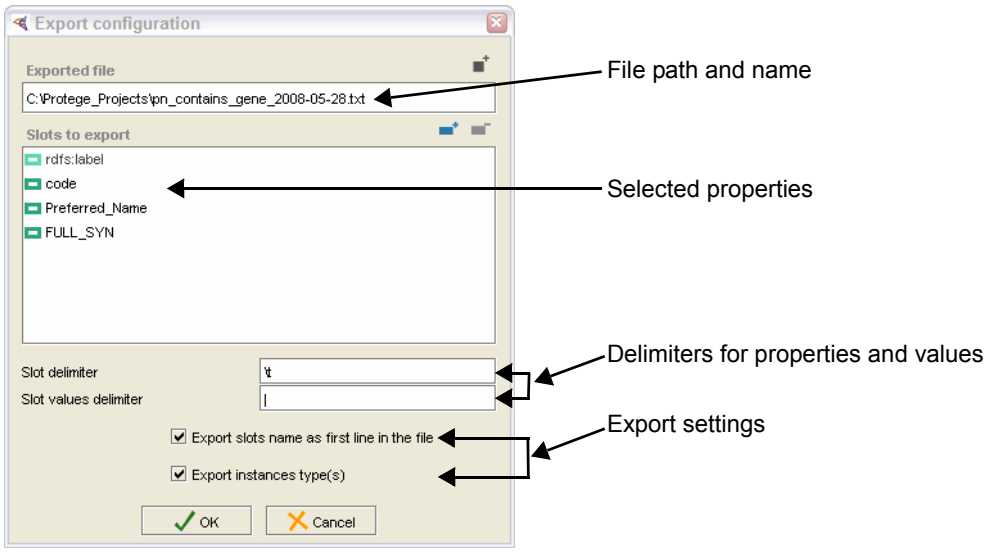


Figure 5.31 Export configuration window with specified settings

8. Click **OK**.

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

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

Saving the File as an Excel Workbook

To save the delimited text file as an Excel workbook, follow these steps:

1. Open Microsoft Excel.
2. Navigate to the saved text file, then open it.

   The Text Import Wizard launches.

3. To set up the workbook, use the following Wizard settings:
   a. On the Step 1 of 3 screen, select **Delimited** as the file type, then click **Next**.
   b. On the Step 2 of 3 screen under **Delimiters**,
      — Ensure that the **Tab** check box is checked.
      — Select the **Other** check box.
      — Type a pipe symbol (|) in the box following the word *Other*.
      — Click **Next**.
   c. On the Step 3 of 3 screen, click **Finish**.

   The delimited data now appear in workbook columns.

4. Save the file in Microsoft Excel (.xls) format.

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

Configuring the Advanced Query Tab

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

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

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

Once you make these changes, the default query on the **Advanced Query** tab should resemble *Figure 5.32*.

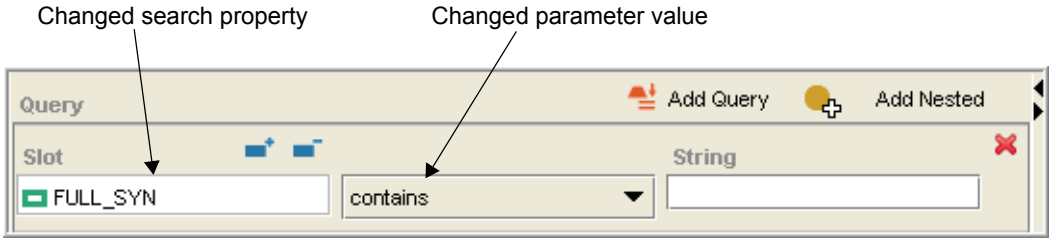


Figure 5.32 Default query with new search property and search parameter

Changing the Default Search Property

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

To change the default search property, follow these steps:

1. Select the following menu command: **Project** > **Configure**.
2. In the **Configure file** window, click the **Property Files** tab.
3. Ensure that the **protege.properties** tab is selected.

   The tab columns 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 5.33*).

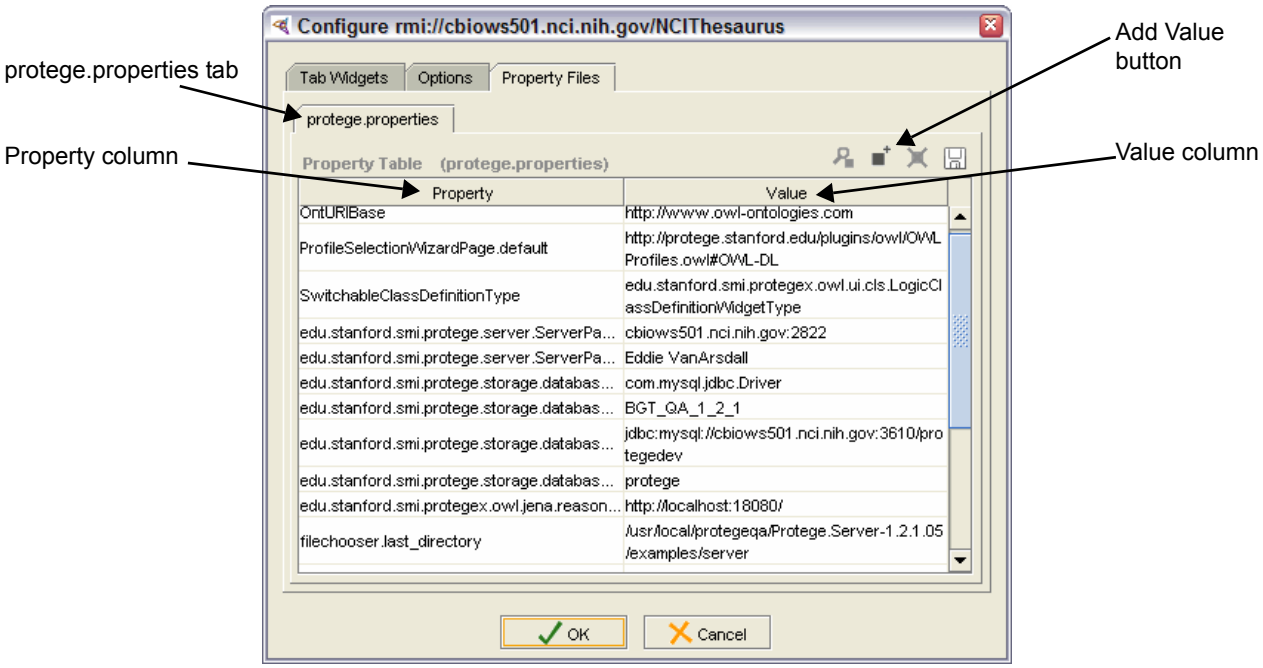


Figure 5.33 Configure File window with protege.properties tab selected

4. Click the **Add Value** button (upper right area, second from left).

   An editable text field appears in the **Property** column (*Figure 5.34*).

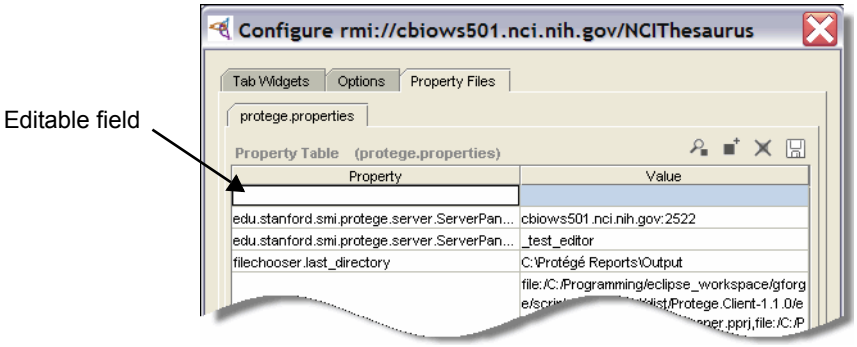


Figure 5.34 Configure File window - Property Files tab with editable field

5. 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.

6. 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.

7. In the **Value** field, type `FULL_SYN`.

8. Press TAB to move the cursor out of the field.

9. Leave the **Configure file** window open for the next section.

Changing the Default Search Parameter

Prerequisite task: *Changing the Default Search Property* on page 75

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

1. With the **Configure file** window still open, click the **Add Value** button .

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

2. In the **Property** field, type the following exactly as shown:
   `query_plugin.default.search_type.String`

3. Press TAB to enable the empty field in the **Value** column on the right.

   This step activates the **Value** field and moves the new property-value combination to the bottom of the list.

4. In the **Value** field, type `contains`.

   The **Configure file** window should now resemble *Figure 5.35* on page 77.

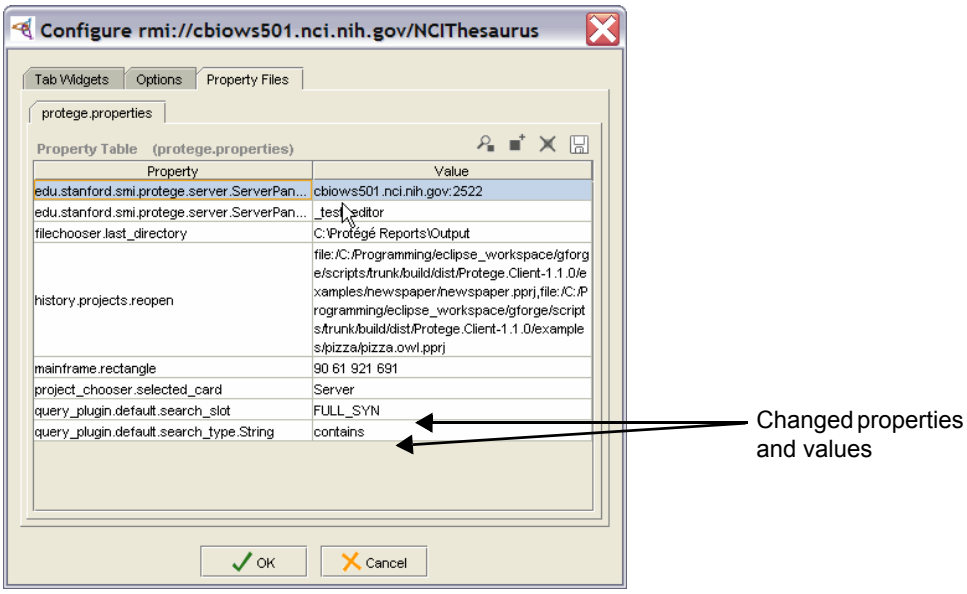


Figure 5.35 Configure File window with new search property and parameter

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

Verifying Search Configuration Changes

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** subtab.

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.

Understanding Where Search Properties and Values Are Stored

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

```
query_plugin.default.search_slot=Preferred_Name

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

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

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

If you configured your settings as specified in *Configuring the Advanced Query Tab* on page 75, 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.

Table 5.2 lists the possible data types with the property name and accepted values for each.

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

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

Table 5.2 Search data types implemented by SMI

From Here...

Working with Classes: The Basics on page 81

CHAPTER

6

WORKING WITH CLASSES: THE BASICS

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

Topics in this Chapter

- *About Classes* on this page
- *Examining Class Information* on page 84
- *Reviewing Changes As You Work* on page 88
- *Creating a Class* on page 89
- *Treeing a Class* on page 92
- *Viewing Classes Connected by part_of Relations* on page 97

About Classes

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

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

Class Identifiers

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

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

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

Class Relationships

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

Class Properties

Although *Working with Properties* (page 101) explains how to add and edit properties, this chapter introduces you to properties because they are integral to working with classes. You also need to understand the subtabs that represent properties when you create and tree classes.

Essentially, properties describe a class:

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

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

Guidelines for Writing Definitions

When writing definitions, observe the rules in this section.

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.

Quoted Definitions

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

Rewritten and Multiple-Source Definitions

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

Definitions Using Journal Articles as Sources

Cite the source as follows:

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

Definitions Using Websites as Sources

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

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

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

Definitions from Other Sources

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

Note: You may sometimes need to add descriptive information for a class when the information isn't actually a definition. For example, you may need to add instructions to a coder regarding how and when a class will be applied, notes to another editor regarding why the class is treed in a specific location, or notes about any additional information that is needed. In this case, you will use an Editor's Note or Design Note. These are used in *Using the PreMerge Subtab to Flag Classes for a Merge* on page 129 and *Using the PreRetire Subtab to Flag a Class for Retirement* on page 139.

Examining Class Information

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

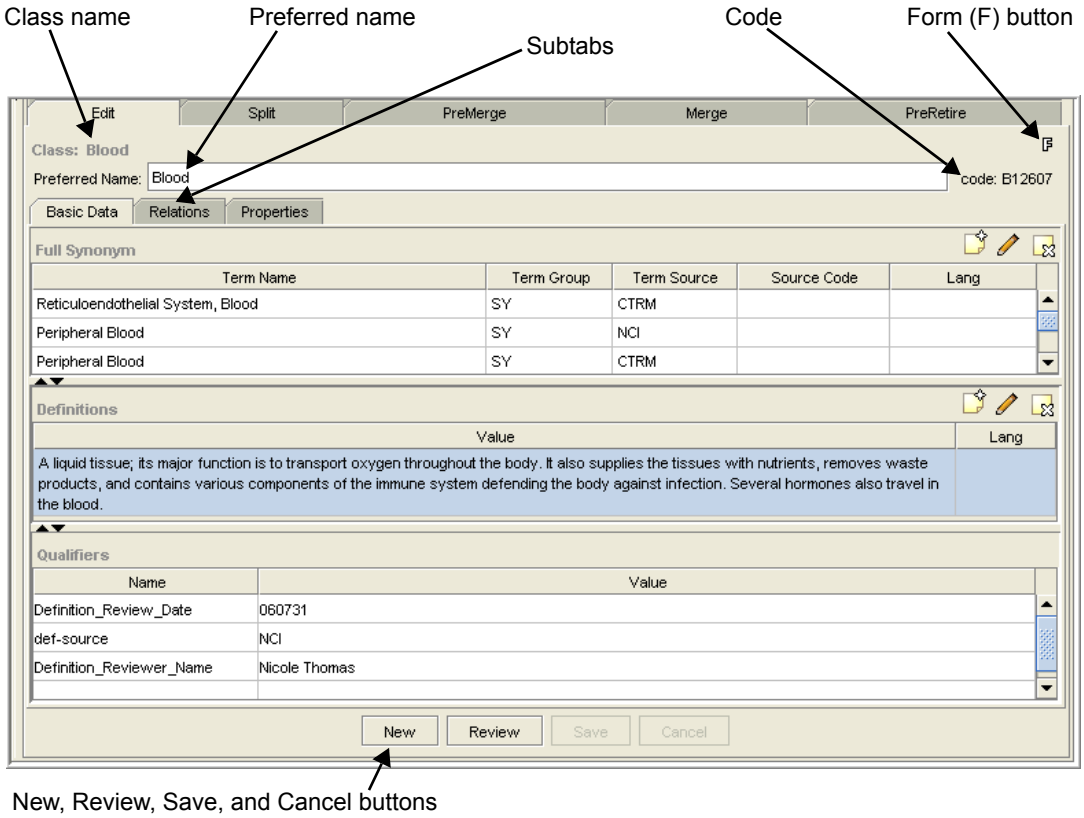


Figure 6.1 Edit subtab

Table 6.1 describes the main features of the **Edit** subtab.

Feature	***Description***
Class Name, Preferred Name, and Code	Collectively known as the header area.
Form button	Used by server administrators to set up Protégé projects.
Basic Data, **Relations**, and **Properties** subtabs	Show detailed information about the class. For more information, see the following sections: • *Basic Data Subtab* on page 85 • *Relations Subtab* on page 86 • *Properties Subtab* on page 87

Table 6.2 Edit subtab features

Feature	Description
New button	Creates a new subclass.
Review button	Opens a window that shows changes made to a class, even before you save them. For more information, see *Reviewing Changes As You Work* on page 88.
Save button	Saves your changes.
Cancel button	Discards unsaved changes.

Table 6.2 Edit subtab features (Continued)

Basic Data Subtab

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

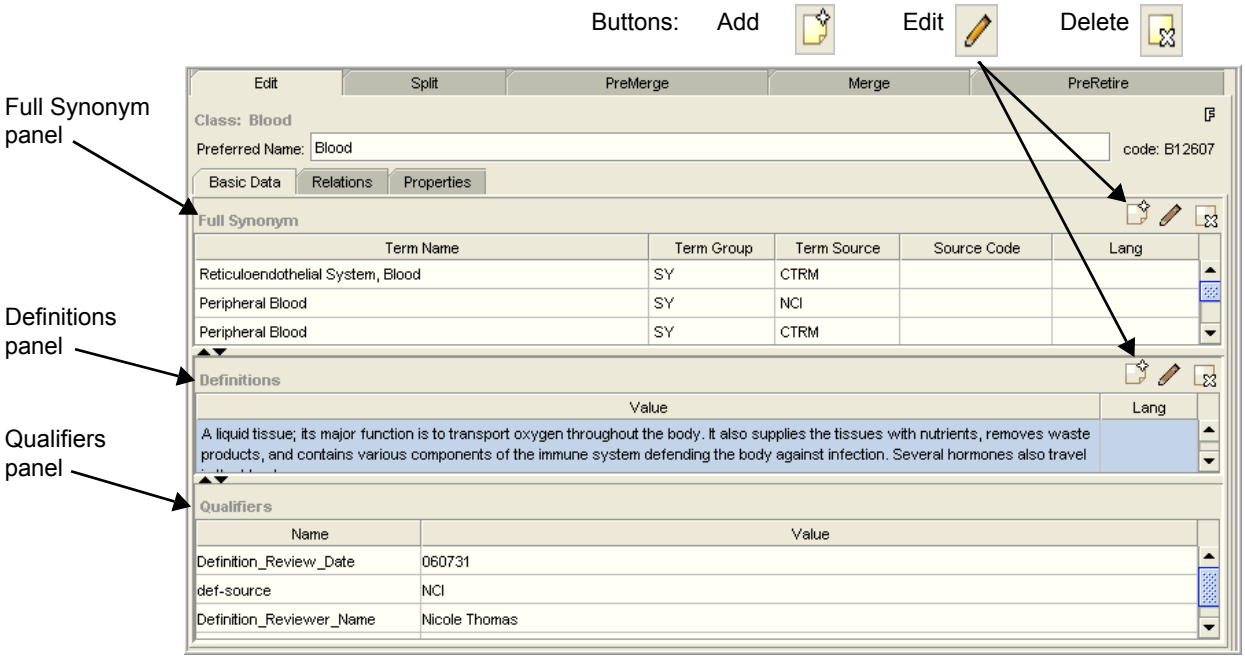


Figure 6.2 Basic Data subtab

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

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

Table 6.3 Basic Data subtab features

Relations Subtab

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

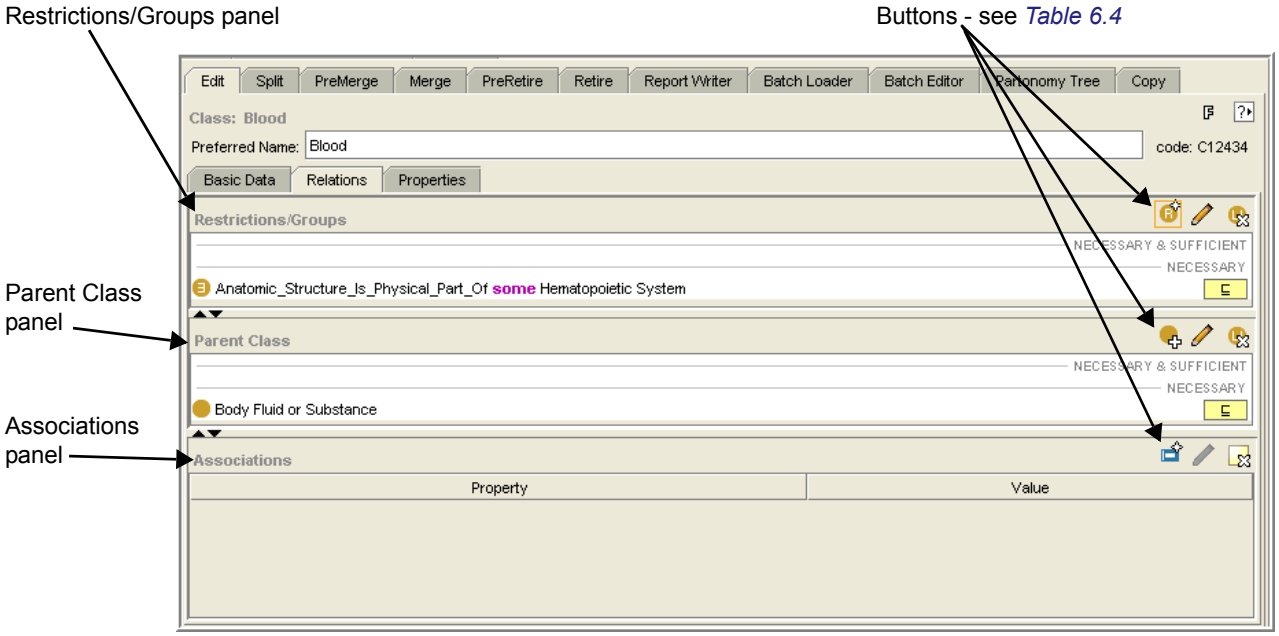


Figure 6.3 Relations subtab

Table 6.4 describes the main features of the Relations subtab.

Feature	***Description***
Restrictions/Groups panel	Shows restriction values.
Add, **Edit**, and **Delete Restriction/Group** buttons	Used to create, edit, or delete a restriction or group.
Parent Class panel	Shows classes that are parents of the selected class.
Add, **Edit**, and **Delete Parent Class** buttons	Used to create, edit, or delete a superclass.
Associations panel	Displays object-valued properties.
Add, **Edit**, and **Delete Object-Valued Property** buttons	Used to create, edit, or delete an object-valued property.

Table 6.4 Relations subtab features

Properties Subtab

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

Note: Additional properties are available on the **Basic Data** subtab (page 85).

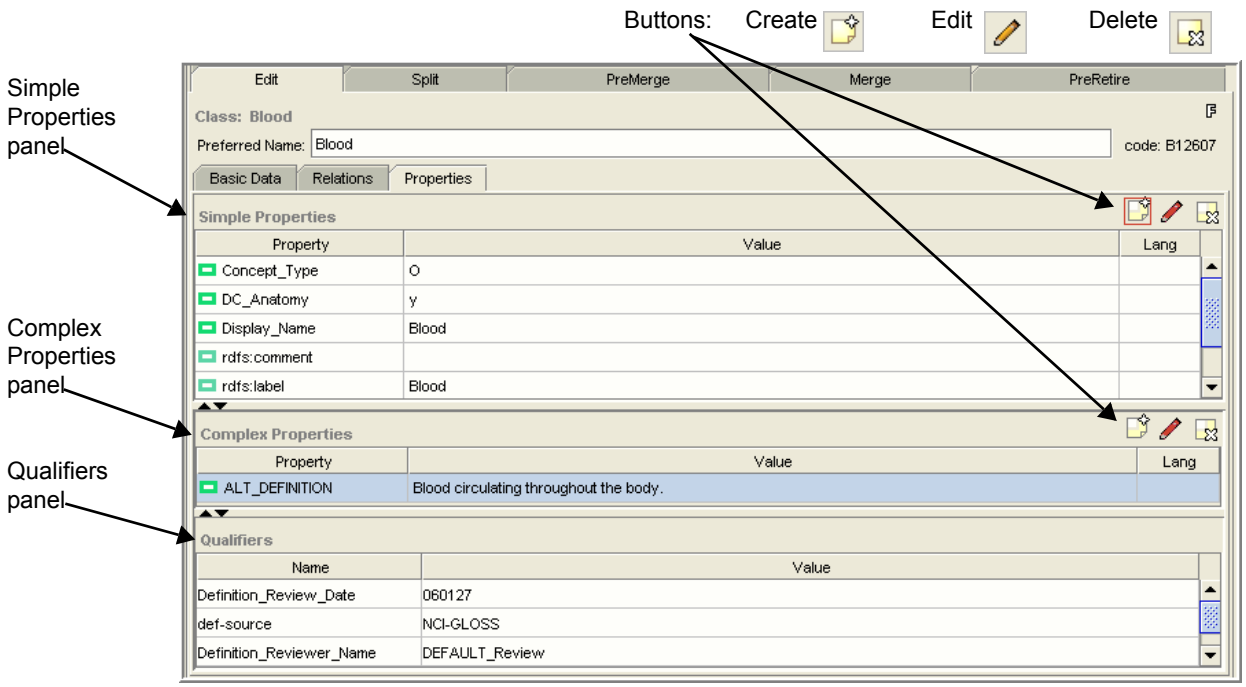


Figure 6.4 Properties subtab

Table 6.5 lists the main features of the **Properties** subtab.

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

Table 6.5 Properties subtab features

Reviewing Changes As You Work

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

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

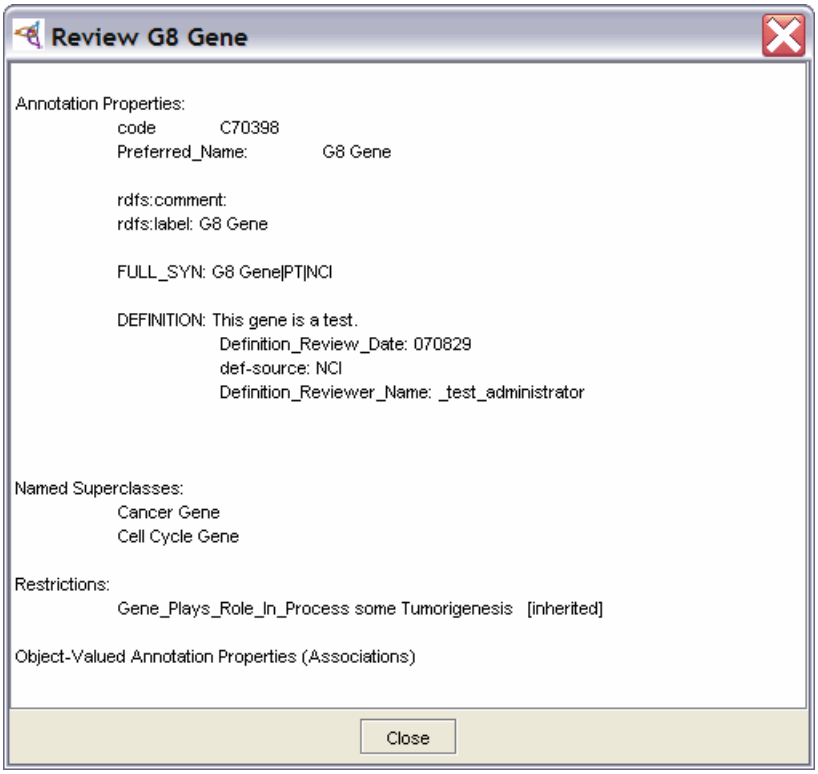


Figure 6.5 Review window

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

Creating a Class

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

Tip: For more information about searching in Protégé, see *Chapter 5, Searching,* on page 53.

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

To create a class, follow these steps:

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

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

2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Basic Data** subtab.
4. Click the **New** button at the bottom of the subtab (*Figure 6.6*).

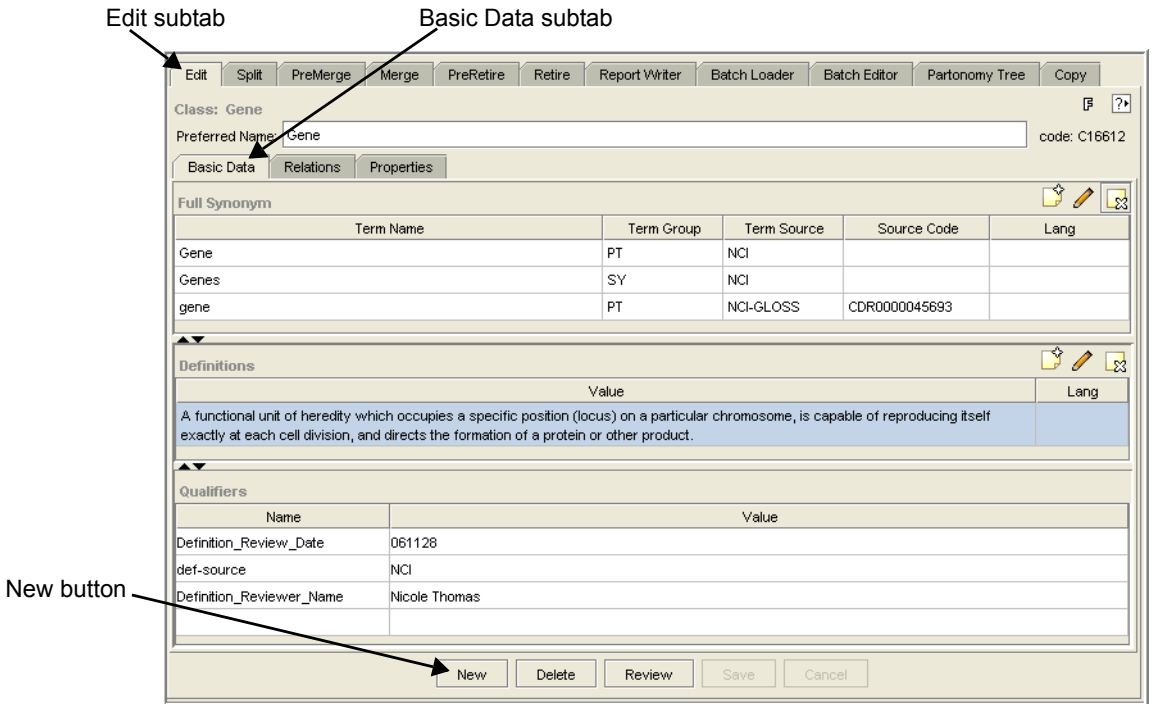


Figure 6.6 Creating a new class

The **Create Subclass** window opens.

5. Click the **Select Superclass** button on the right (*Figure 6.7*).

This step enables you to select a parent class.

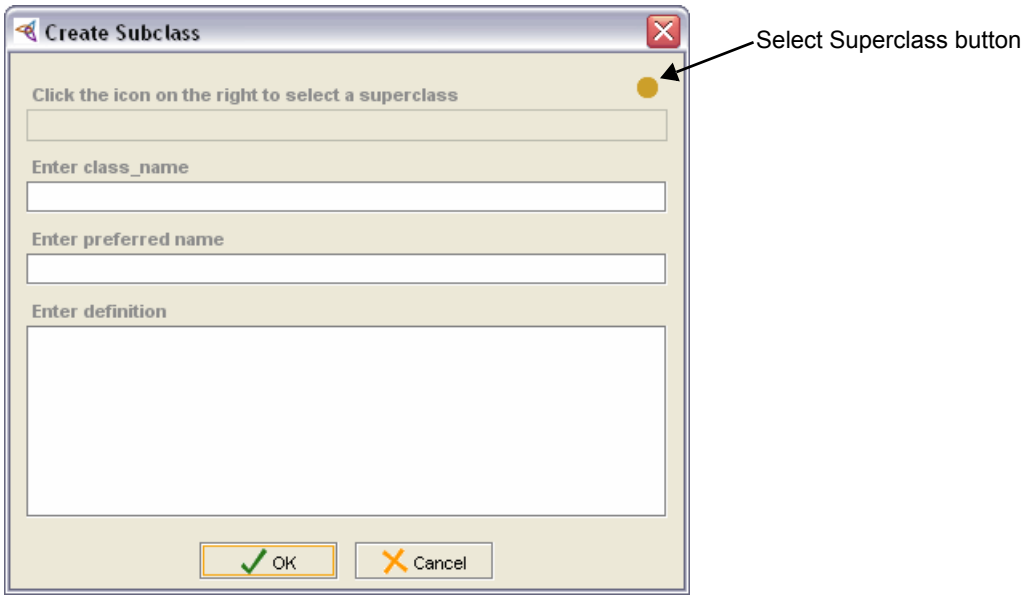


Figure 6.7 Create Subclass window

The **Select a superclass** window opens (*Figure 6.8*).

6. Select or search for a class.

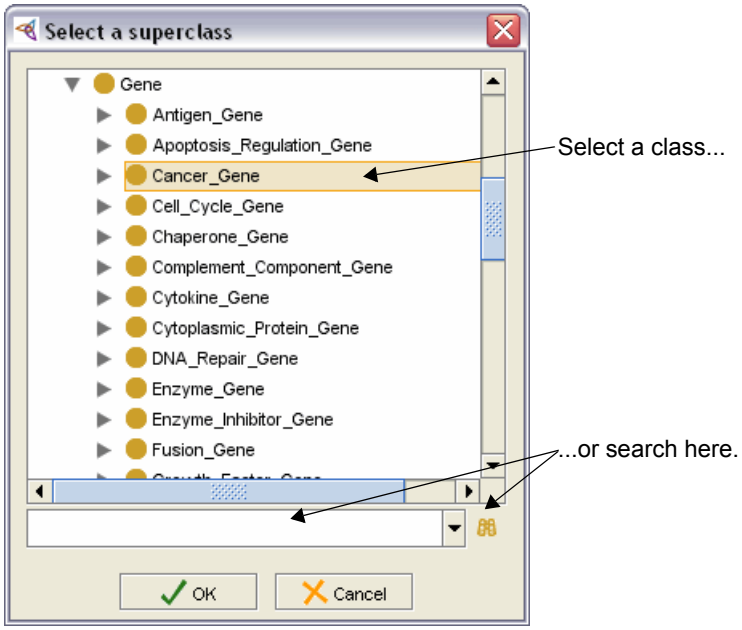


Figure 6.8 Select a superclass window

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

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

   The name of the selected class now appears in the upper part of the window (*Figure 6.9*).

8. Enter a class name, preferred name, and definition.
9. (Optional) If you wrote the definition in a word processor, copy and paste it into the **Enter definition** field.
10. Verify that the **Create Subclass** window now resembles *Figure 6.9*.

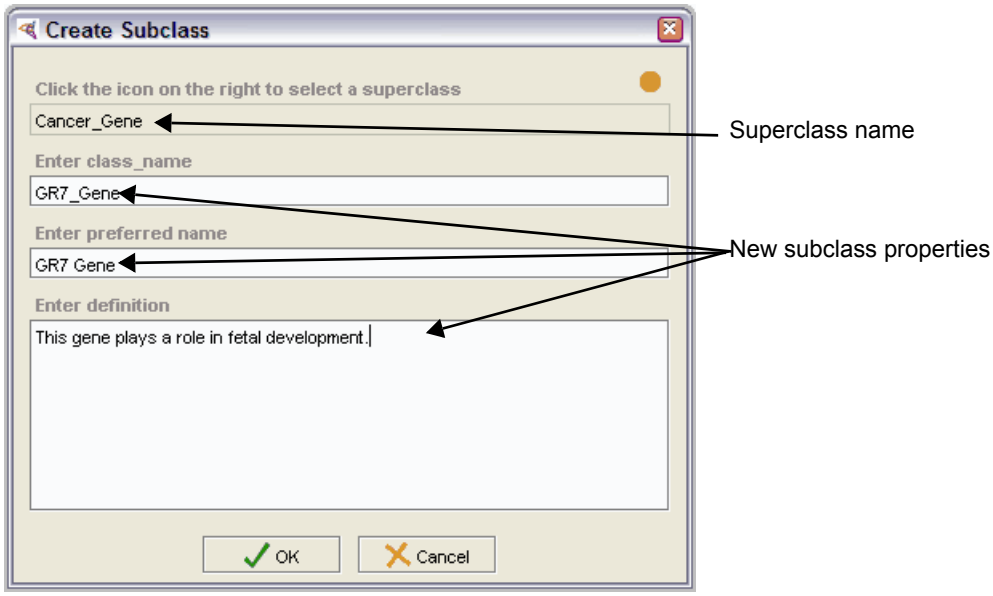


Figure 6.9 Create Subclass window with completed fields

11. Click **OK** to close the **Create Subclass** window.
12. Verify the following results in the main window:
    - The new class is now highlighted in the Class Hierarchy on the left.
    - The **Basic Data** subtab (*Figure 6.10* on page 92), shows the following information:
      - The **Full Synonym** panel shows the *Term Name*, *Term Group*, and *Term Source*.
      - The **Definitions** panel shows the definition.
      - The **Qualifiers** panel shows the *Definition_Review_Date*, *def-source*, and *Definition_Reviewer_Name*.

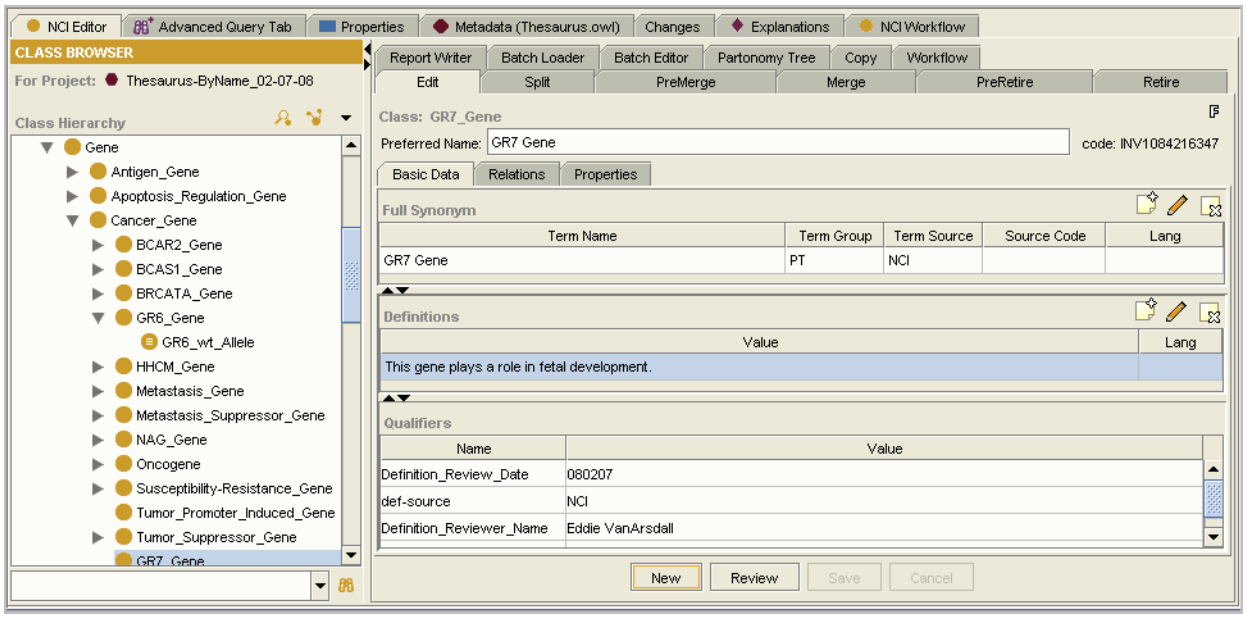


Figure 6.10 New class information

Treeing a Class

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

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

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

Topics in this Section

- *Adding a Parent Class* on this page
- *Modifying a Parent Class* on page 95
- *Deleting a Parent Class* on page 96

Adding a Parent Class

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

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

1. Using the Class Browser, select the class to which you want to add a parent class.
2. Click the **Edit** subtab on the right if it is not already displayed.
3. Click the **Relations** subtab.
4. Select a class in the Parent Class panel.
5. Click the **Add Named Class** button (*Figure 6.11*).

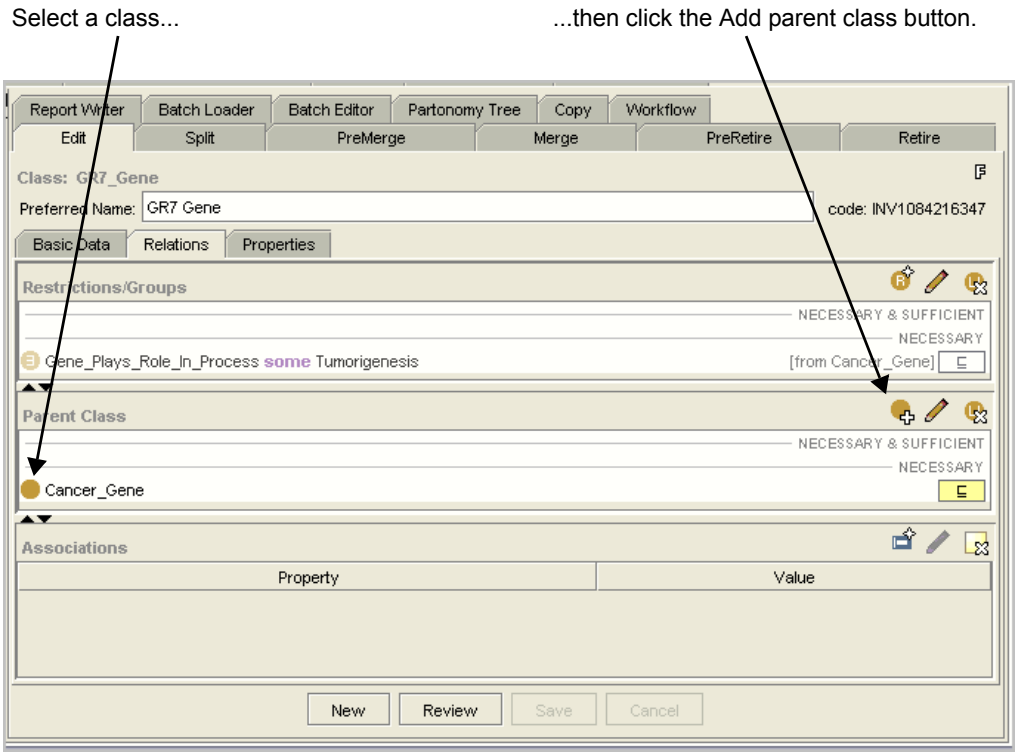


Figure 6.11 Edit subtab - Parent Class panel

The **Select a class** window opens (*Figure 6.12* on page 94).

6. Select or search for a class.

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

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

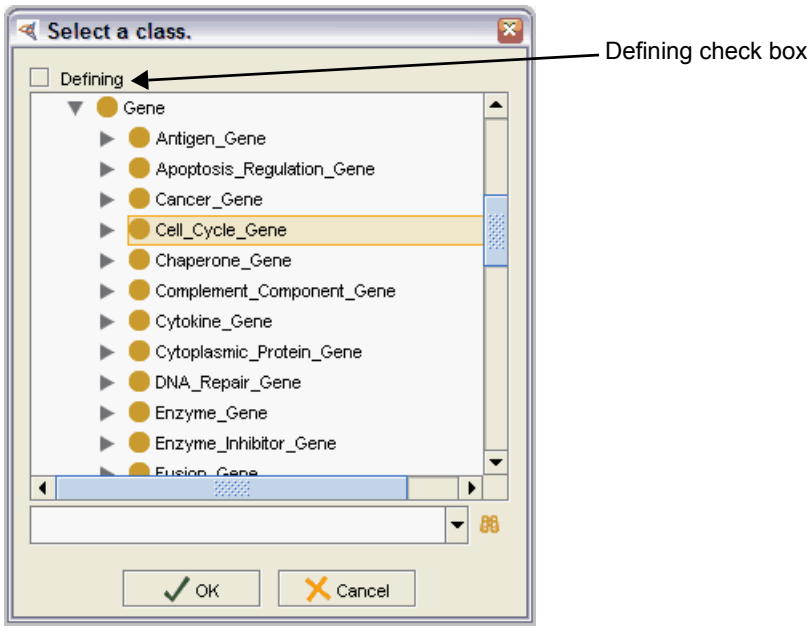


Figure 6.12 Select a class window

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

   The **Parent Class** panel now shows the new class (*Figure 6.13*).

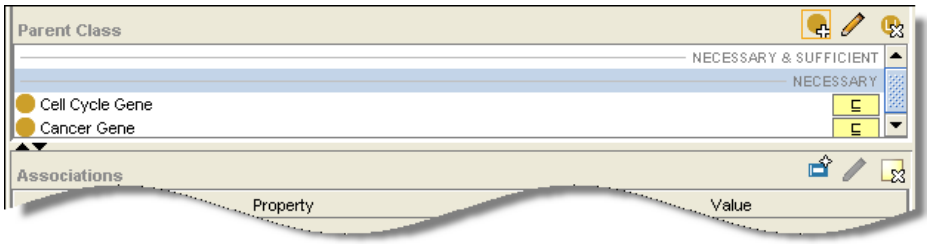


Figure 6.13 Parent Class panel with new class

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

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

   A message confirms that the class has been saved successfully.

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

Modifying a Parent Class

To modify a parent class, follow these steps:

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

   The **Modify Named Superclass** window opens (*Figure 6.14*).
6. Click the **Select a superclass** button on the right.

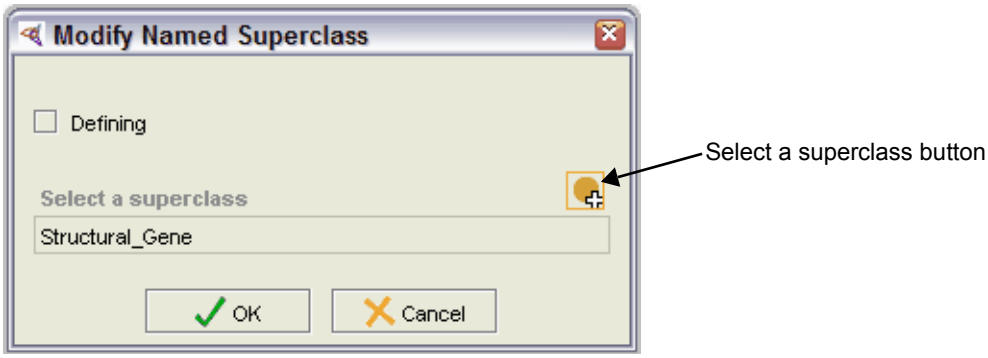


Figure 6.14 Modify Named Superclass window

7. Select or search for a class.

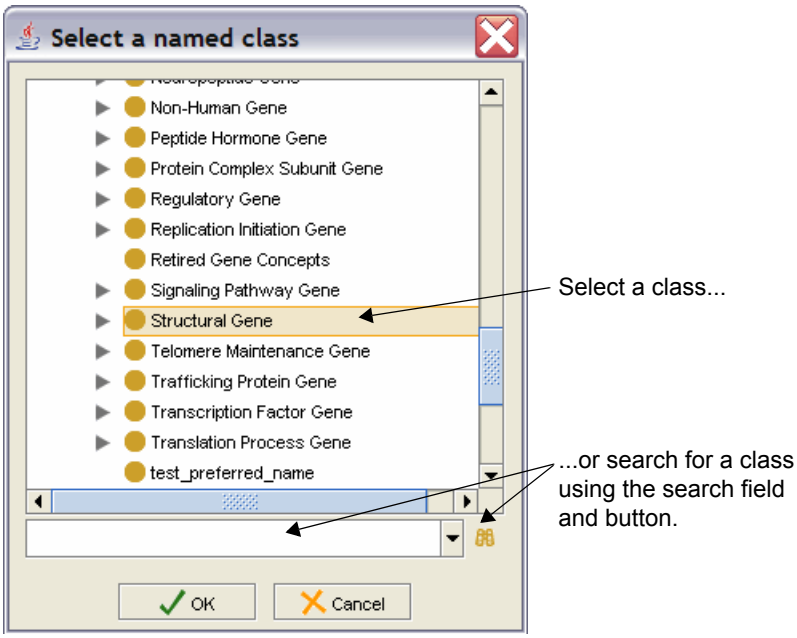


Figure 6.15 Select a named class window

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

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

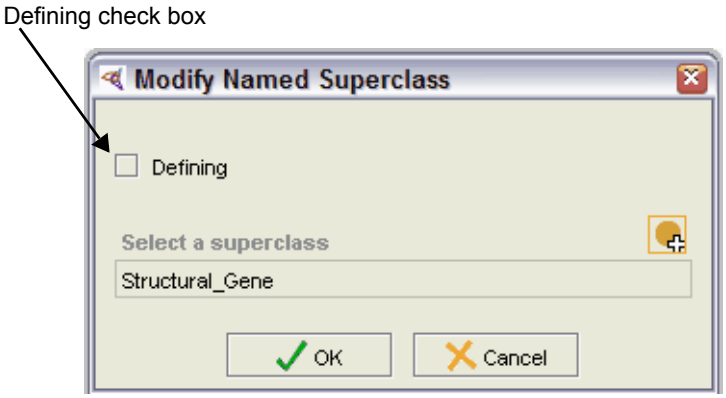


Figure 6.16 Modified parent class

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

    A message confirms that the class has been saved successfully.

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

Deleting a Parent Class

To delete a parent class, follow these steps:

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

   Note: If only one parent class shows in the list, you cannot delete it.

5. Click the **Delete Selected Row** button in the upper right area of the panel.
6. Click **Yes** when the confirmation message appears.

   The selected class is removed from the **Parent Class** panel.

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

   A message confirms that the class has been saved successfully.

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

Viewing Classes Connected by part_of Relations

About the Partonomy Tree Subtab

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

Figure 6.17 shows the layout of the **Partonomy Tree** subtab.

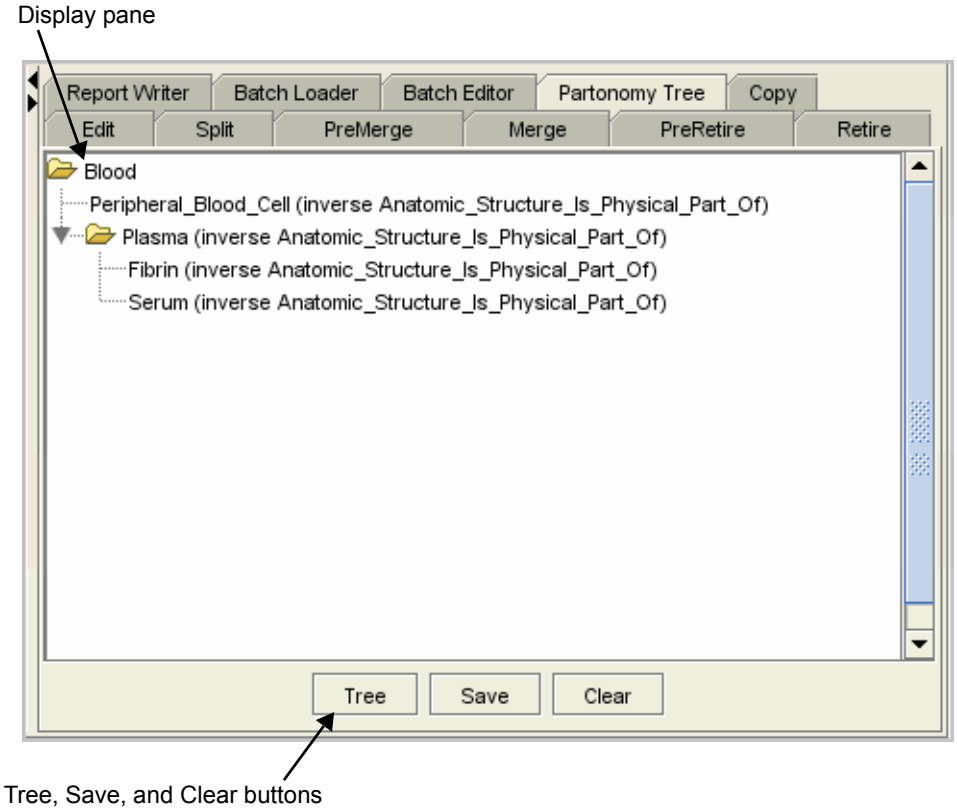


Figure 6.17 Partonomy Tree subtab

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

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

Table 6.6 Partonomy Tree subtab features

Generating a Partonomy Tree

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

1. Using the Class Browser, select a root class.
2. Click the **Partonomy Tree** subtab.
3. Click the **Tree** button at the bottom of the subtab (*Figure 6.18*).

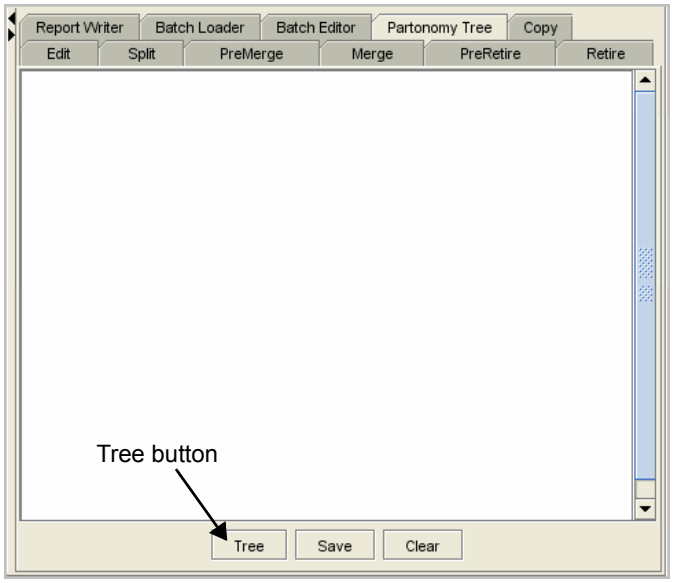


Figure 6.18 Partonomy Tree subtab

The **Select Transitive Properties** window opens (*Figure 6.19*).

4. Select one or more restriction names.

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

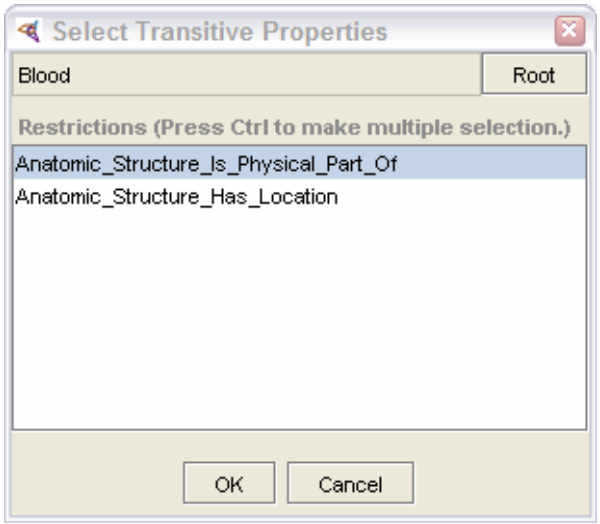


Figure 6.19 Select Transitive Properties window

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

The partonomy tree appears in the tree view area of the subtab (*Figure 6.20*).

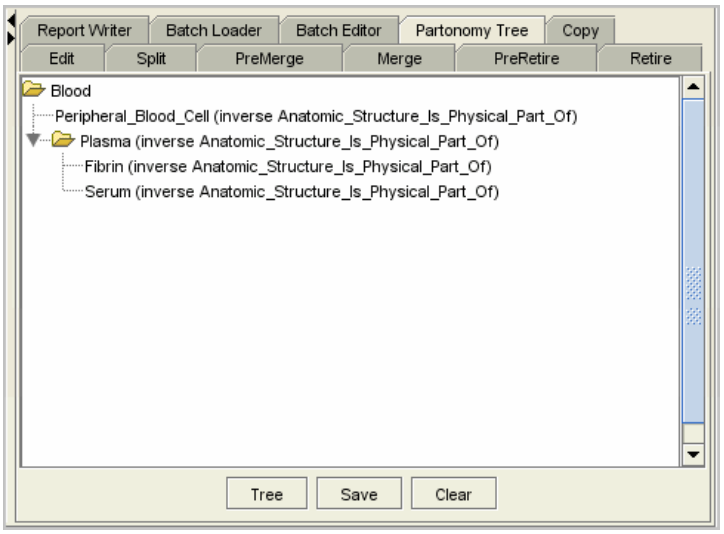


Figure 6.20 Partonomy tree for Blood class

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.

From Here...

Working with Properties on page 101

CHAPTER

7

WORKING WITH PROPERTIES

This chapter explains how to use the **Basic Data**, **Relations**, and **Properties** subtabs to create and manage properties.

Topics in this Chapter

- *Asserting Annotation Properties* on this page
- *Asserting Relations* on page 106
- *Adding an Association* on page 117
- *Editing Two Classes at the Same Time* on page 120

Asserting Annotation Properties

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

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

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

Note: In this chapter, you will work with the **Edit** subtab and its three subtabs: **Basic Data**, **Relations**, and **Properties**. For an introduction to these subtabs, see *Examining Class Information* on page 84

Topics in this Section

- *Adding a Full Synonym* on page 102
- *Modifying a Full Synonym* on page 103
- *Deleting a Full Synonym* on page 104
- *Modifying a Definition* on page 104

Adding a Full Synonym

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

To add a full synonym, follow these steps:

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

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

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

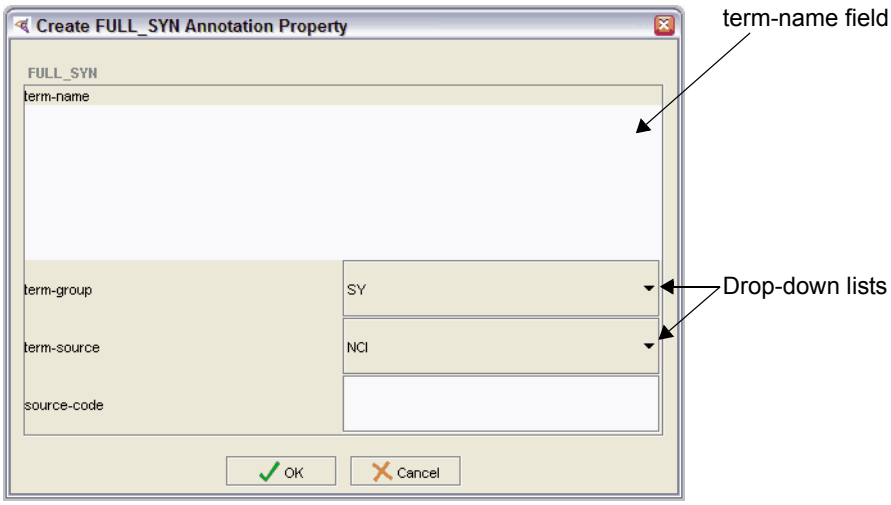


Figure 7.1 Create FULL_SYN Annotation Property window

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

8. The new synonym now appears in the **Full Synonym** list.

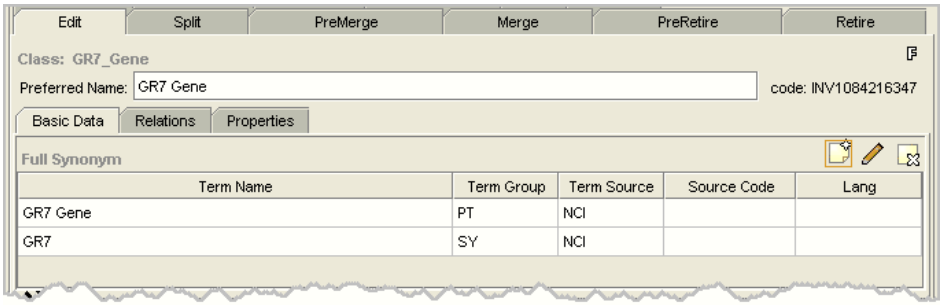


Figure 7.2 New synonym

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

   A message confirms that the class has been saved successfully.

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

Modifying a Full Synonym

To modify a full synonym, follow these steps:

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

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

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

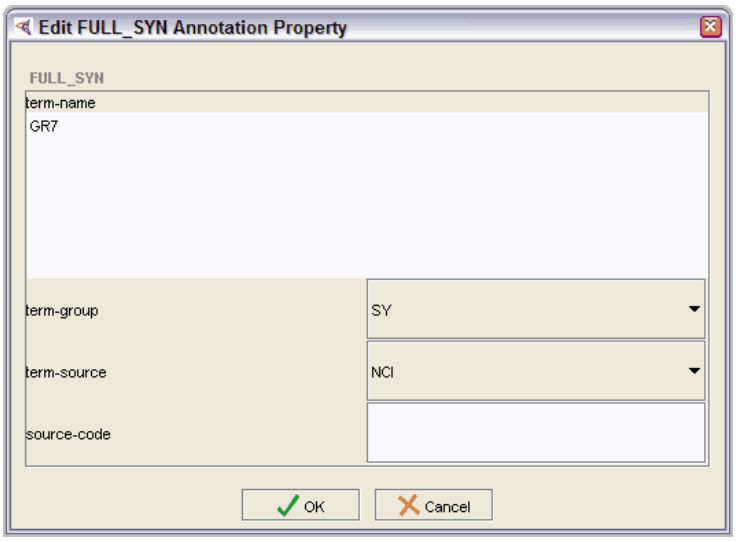


Figure 7.3 Edit FULL_SYN Annotation Property window

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

   A message confirms that the class has been saved successfully.

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

Deleting a Full Synonym

To delete a full synonym, follow these steps:

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

   The selected property is removed from the **Full Synonym** panel.

   Tip: If you deleted a property by mistake, click the **Cancel** button to restore the property.
7. Click the **Save** button to save the change.

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

Modifying a Definition

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

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

   The **Edit DEFINITION Annotation Property** window opens (*Figure 7.4* on page 105).

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

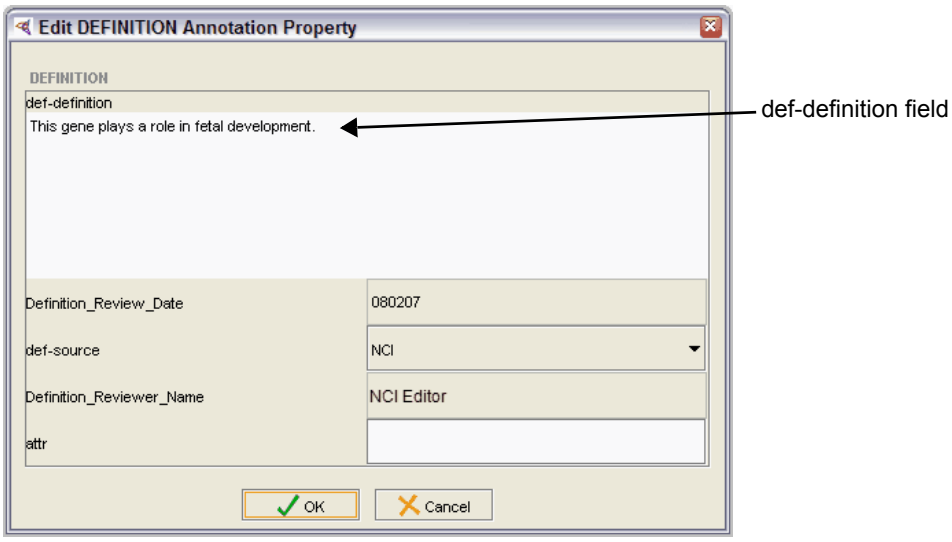


Figure 7.4 Edit DEFINITION Annotation Property window

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

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

Asserting Relations

Topics in this Section

- *Adding a Restriction* on this page
- *Modifying a Restriction* on page 109
- *Deleting a Restriction* on page 112
- *Adding a Role Group* on page 113

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

Adding a Restriction

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

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

   The **Create a Restriction** window opens (*Figure 7.5*).
5. Click the **Create a Role** button, the first of the three buttons in the right side of the window.

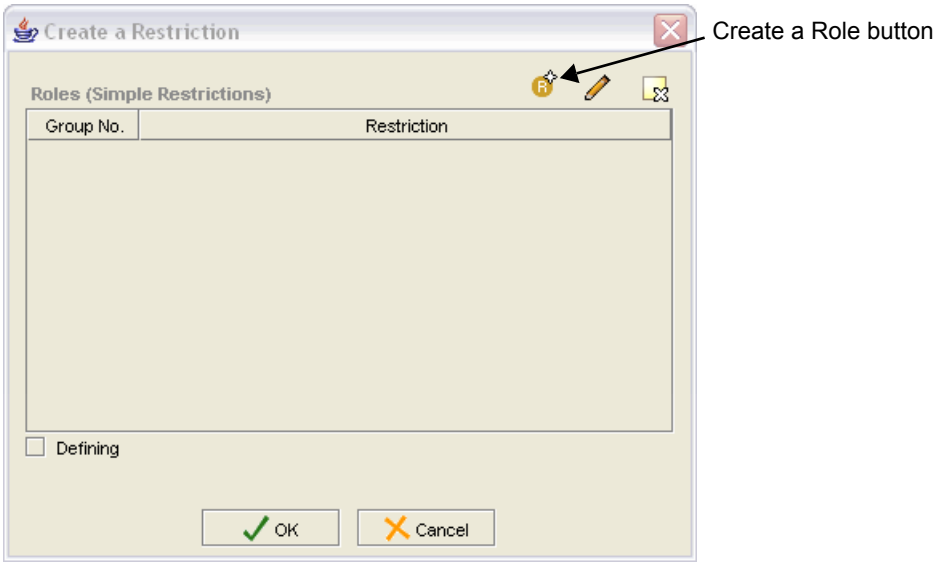


Figure 7.5 Create a Restriction window (initial)

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

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

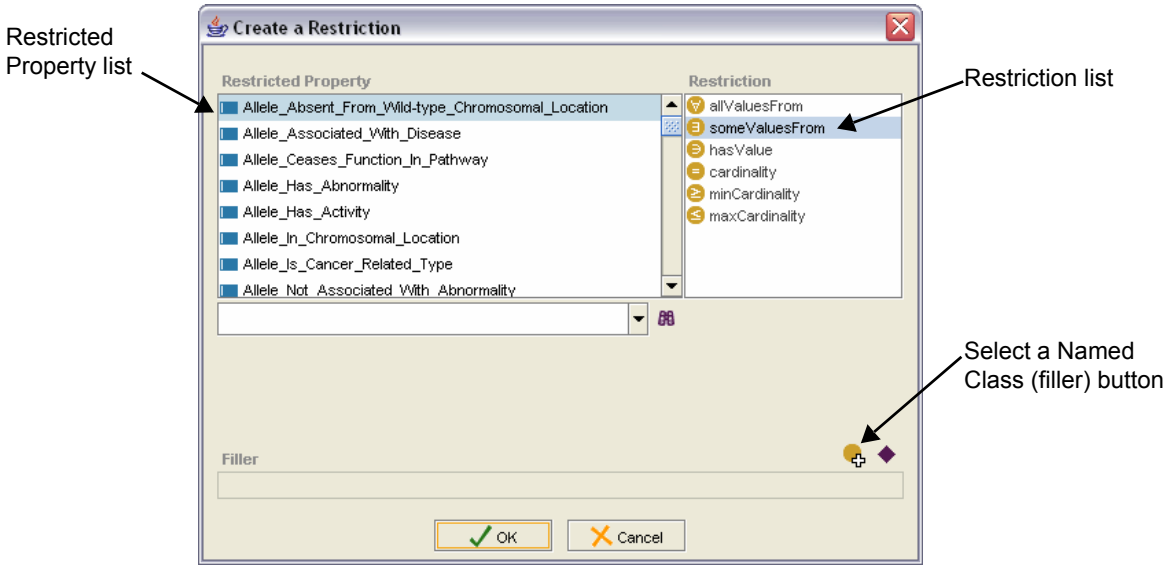


Figure 7.6 Create a Restriction window (second of two windows)

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

9. Select or search for a class.

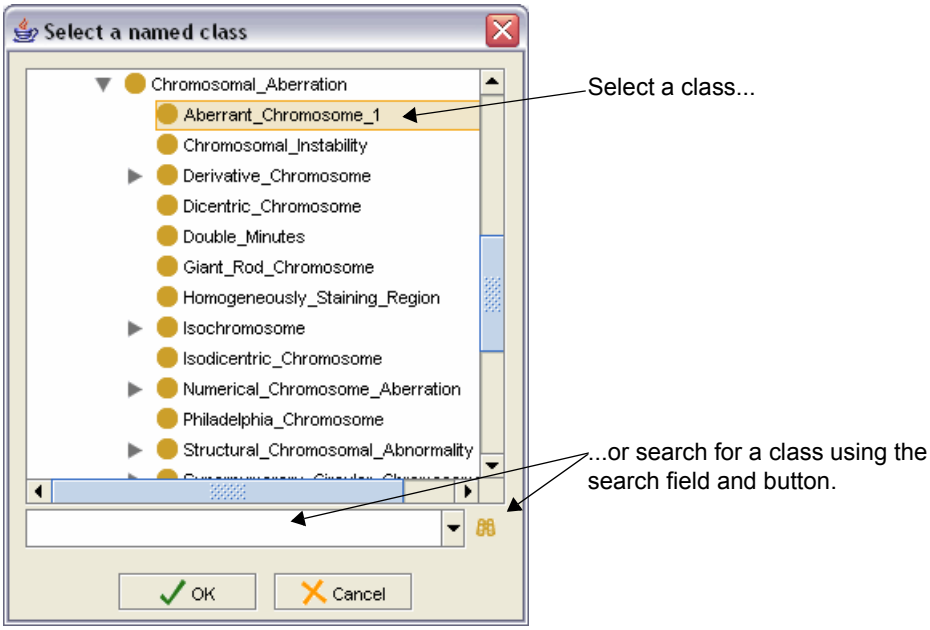


Figure 7.7 Select a named class window

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

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

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

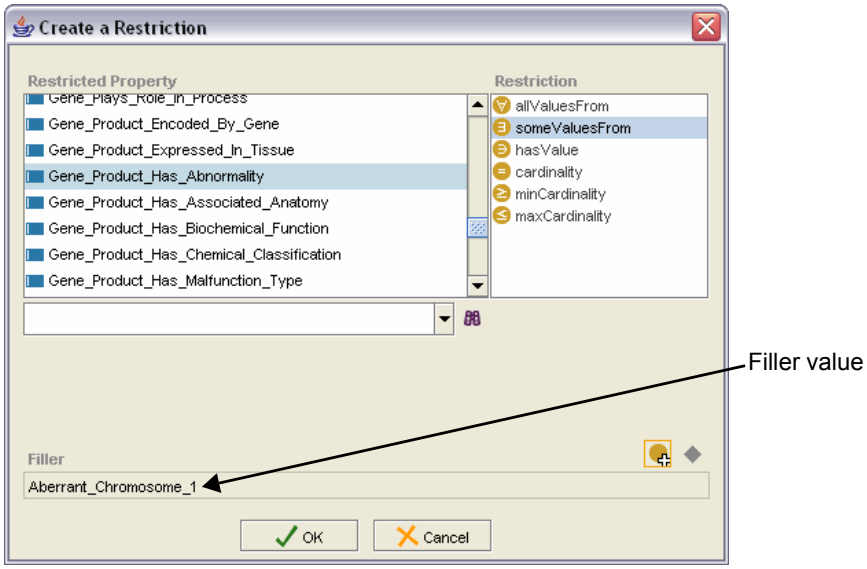


Figure 7.8 Create a Restriction window with Filler value

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

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

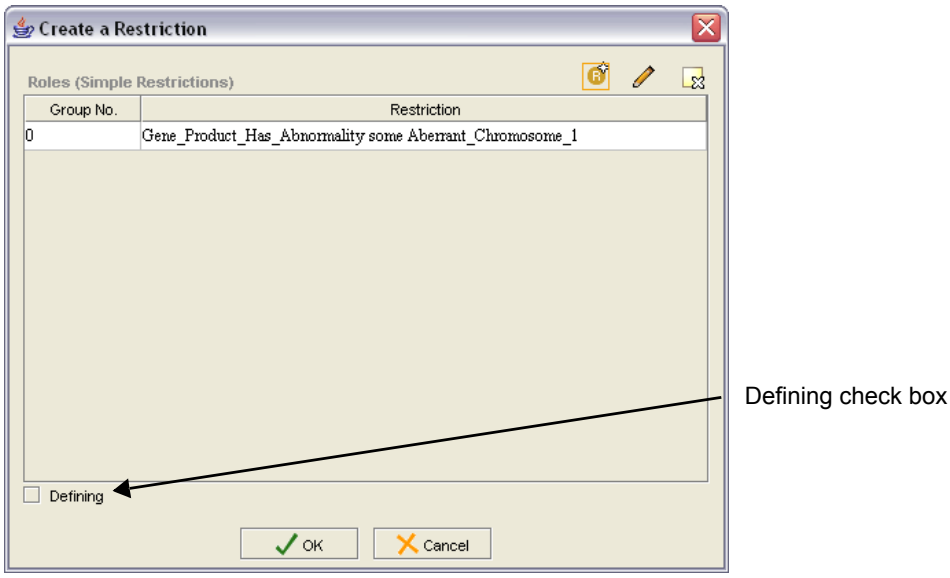


Figure 7.9 Newly added restriction

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

    The new restriction now appears in the **Restrictions/Groups** list.
14. Click the **Save** button to save the change.

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

Modifying a Restriction

To modify a restriction, follow these steps:

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

   Note: Inherited restrictions appear in the lower area of the panel. Even though you can select them, you cannot edit them. If you try, the following message appears: *Cannot modify inherited restriction*.
5. Click the **Edit a restriction/group** button in the upper right area of the panel (*Figure 7.10*).

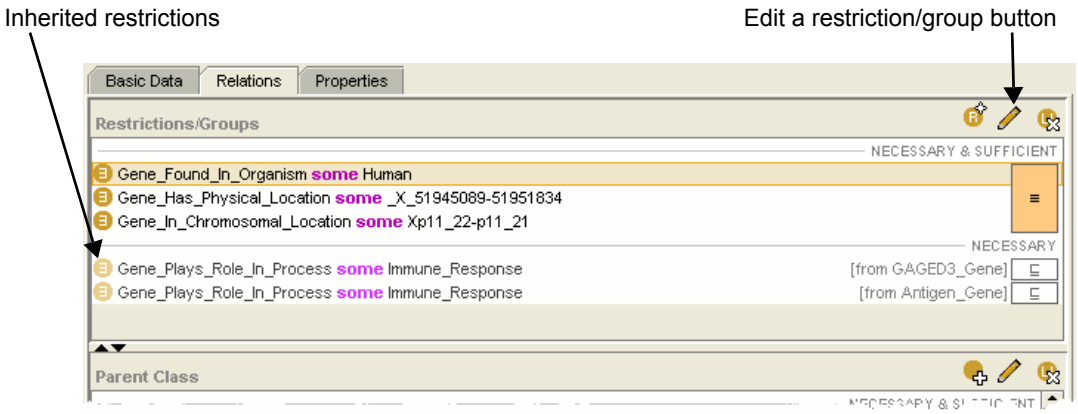


Figure 7.10 Restrictions/Groups panel with inherited restrictions

The **Edit a Restriction** window opens (*Figure 7.11* on page 110).

6. Select the restriction to be modified, even if only one restriction appears in the window.
7. Click the **Modify a role (simple restriction)** button in the upper right area of the window. (See *Figure 7.11*.)

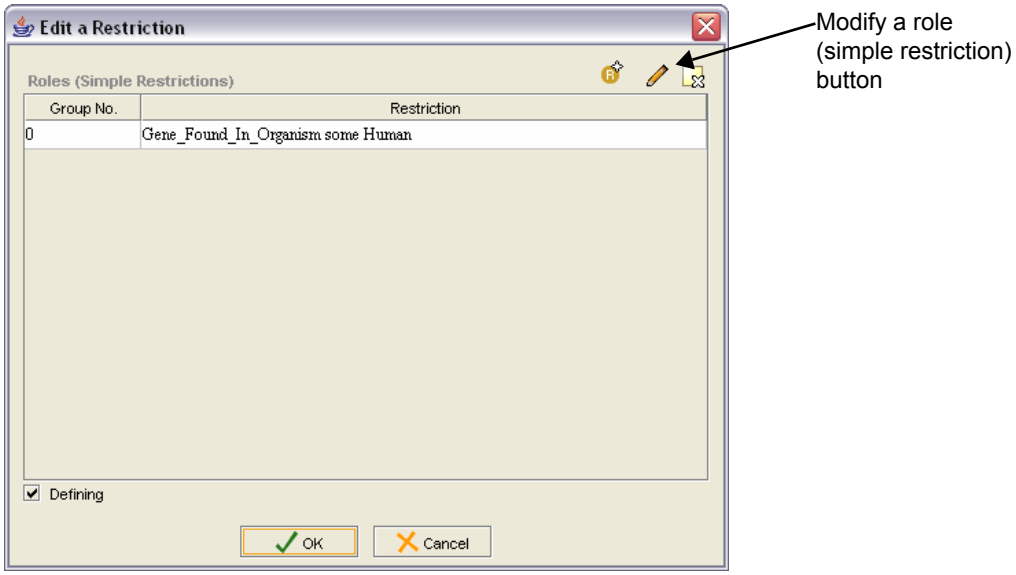


Figure 7.11 Edit a Restriction window

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

8. Select a property in the **Restricted Property** list.
9. Select a modifier in the **Restriction** list (for example, `someValuesFrom`).
10. Click the **Select a Named Class (filler)** button in the lower right area of the window (*Figure 7.12*).

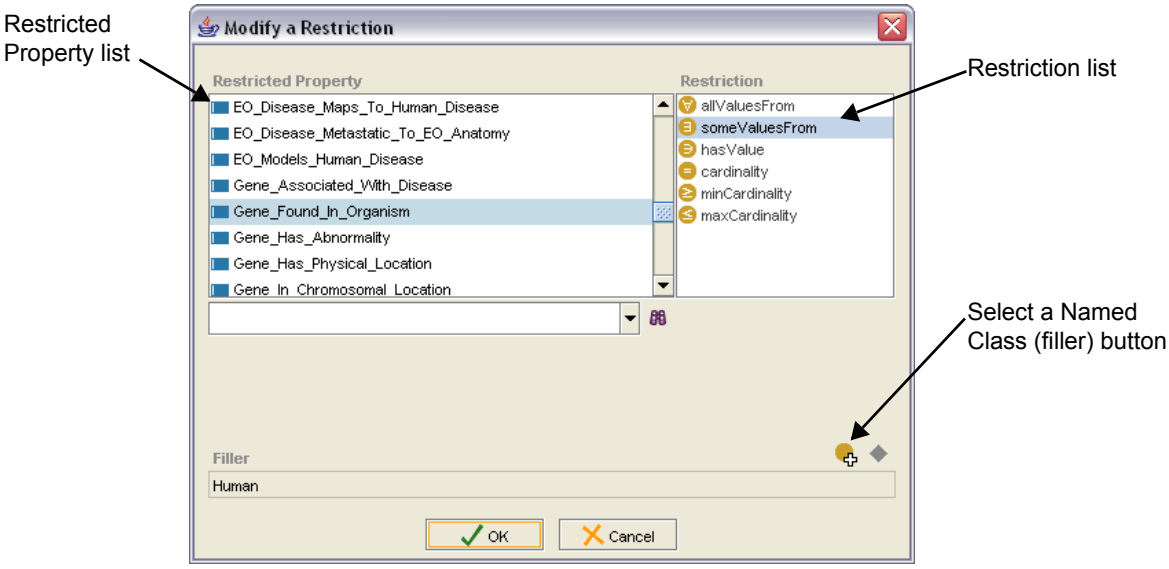


Figure 7.12 Modify a Restriction window

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

11. Select or search for a class.

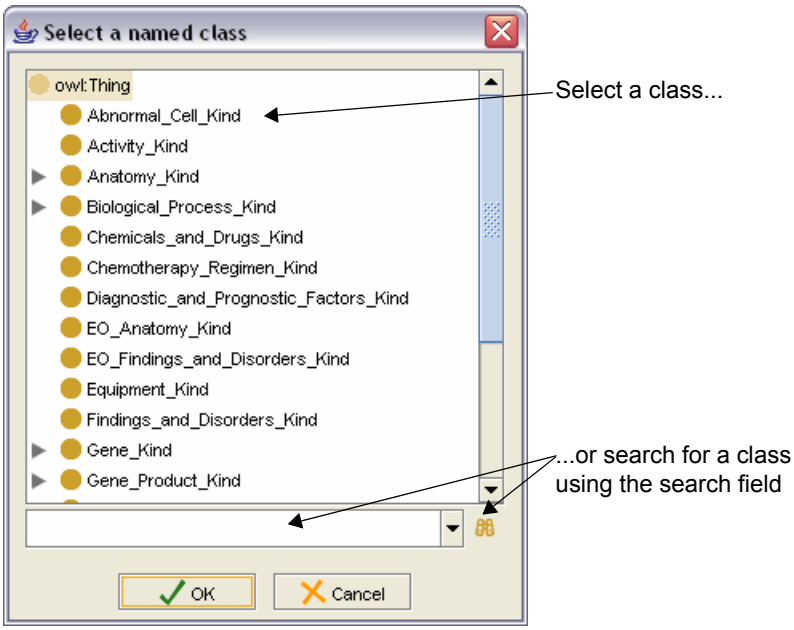


Figure 7.13 Select a named class window

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

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

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

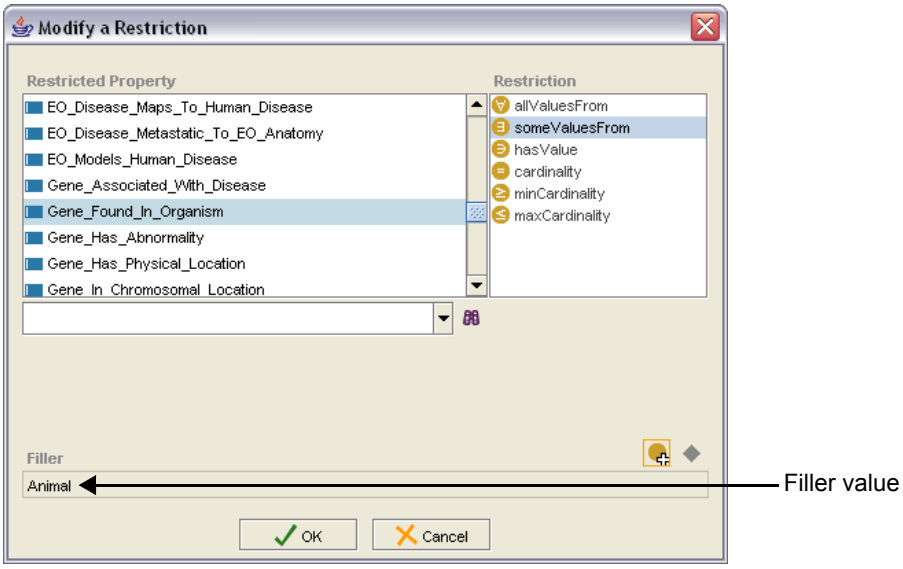


Figure 7.14 Modify a Restriction window with new Filler value

13. Click **OK** to return to the **Edit a Restriction** window. The new restriction now appears in the original window.

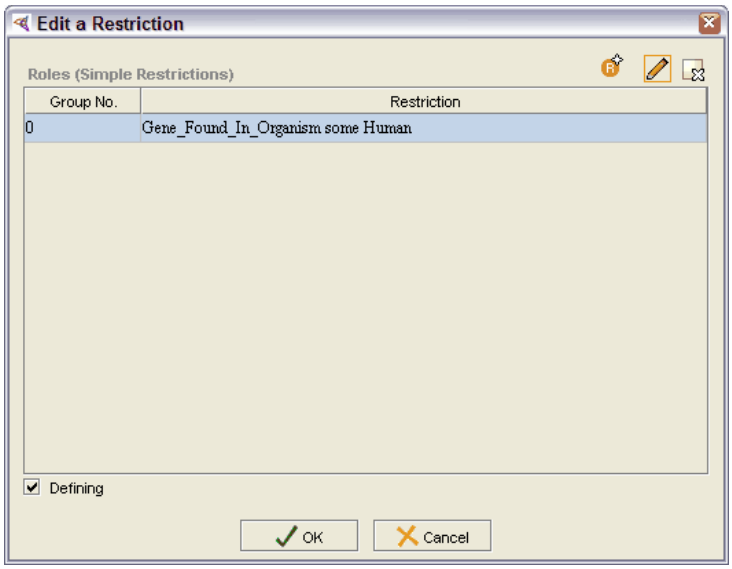


Figure 7.15 Newly modified restriction

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

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

Deleting a Restriction

To delete a restriction, follow these steps:

1. Using the Class Browser, select the class from which you want to delete a restriction.
2. Click the **Edit** subtab if it is not already displayed.
3. Click the **Relations** subtab.
4. Select the restriction to be deleted on the **Restrictions** panel.
5. Click the **Delete selected row** button , 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 save the change.

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

Adding a Role Group

To add a role group, follow the steps in this section. For more information about role groups, see *OWL Class Descriptions and Anonymous Classes* on page 18.

Adding Multiple Restrictions

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

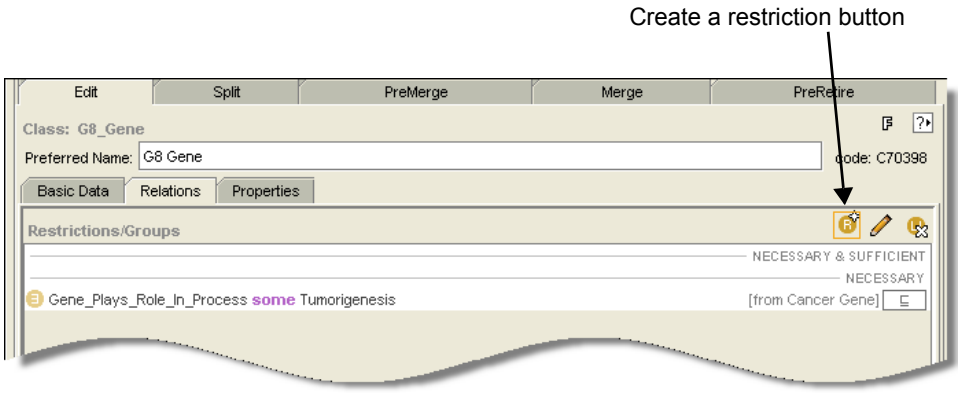


Figure 7.16 Edit sub > Relations subtab > Restrictions panel

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

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

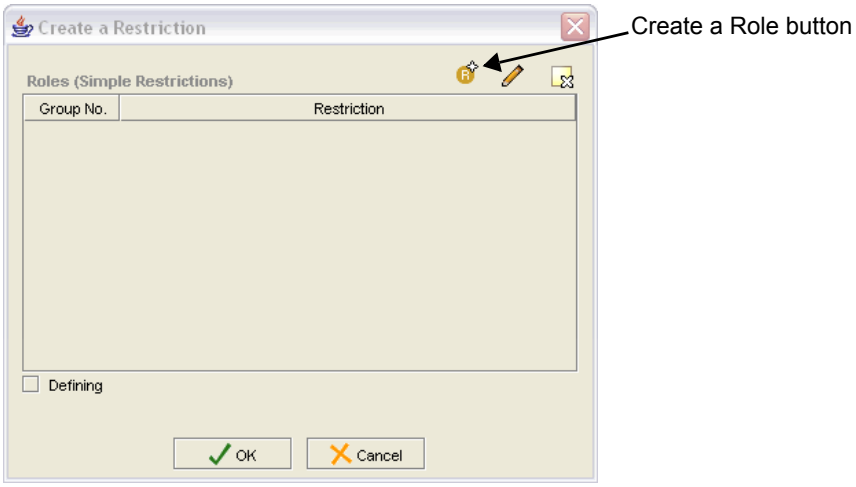


Figure 7.17 Create a Restriction window (first of two windows)

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

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

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

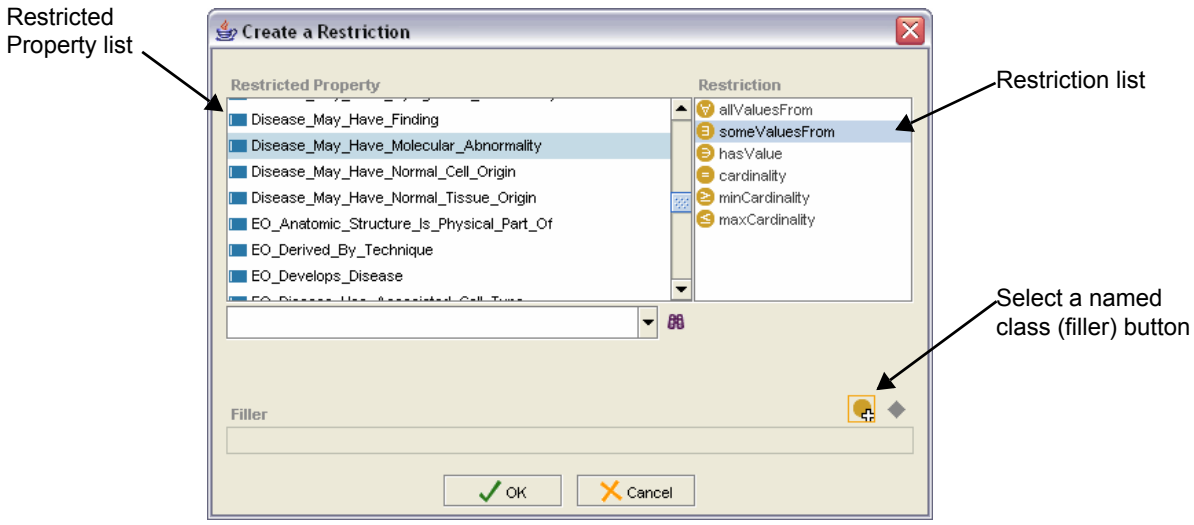


Figure 7.18 Create a Restriction window (second of two windows)

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

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

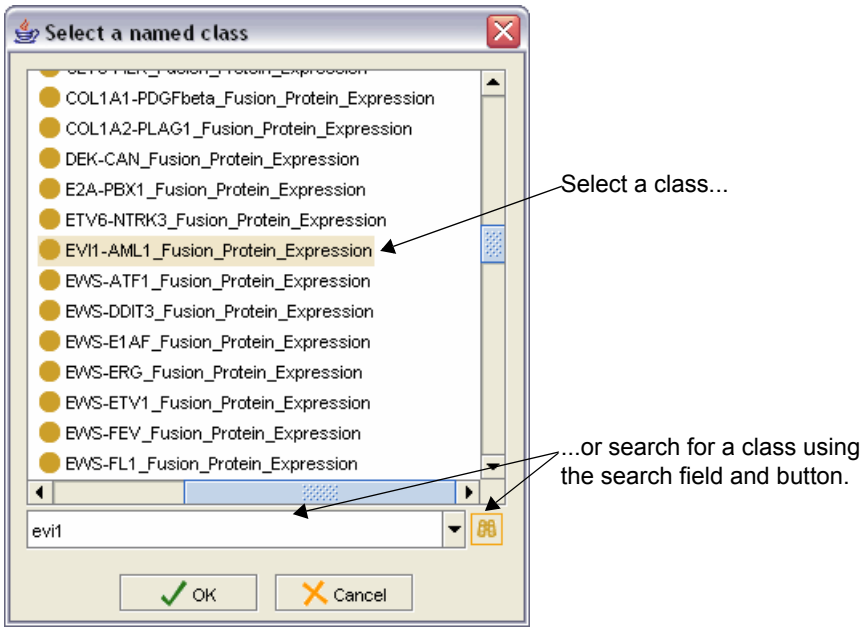


Figure 7.19 Select a named class window

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

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

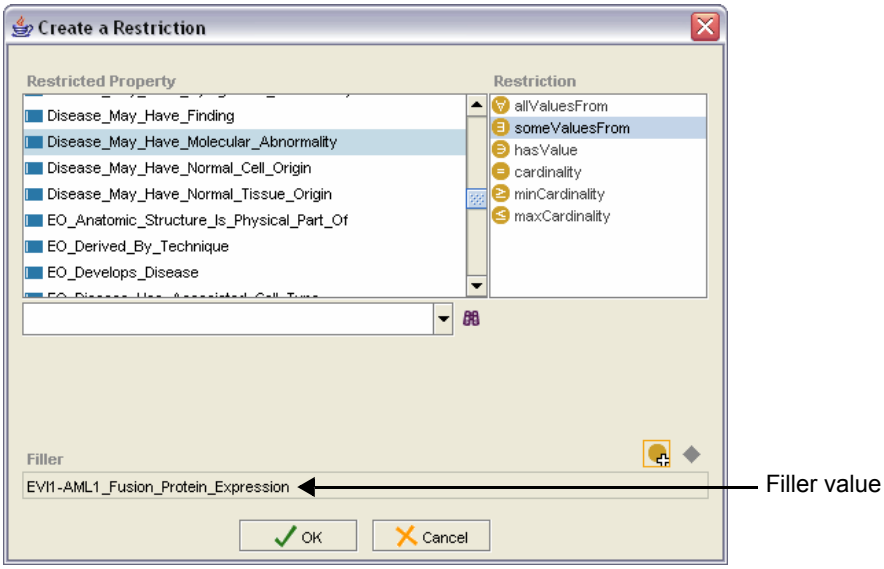


Figure 7.20 Create a Restriction window with Filler value

10. Repeat Steps *6.* through *9.* to add more restrictions.
11. Once you are finished, click **OK** to return to the original window.

    The window now shows a group of restrictions.

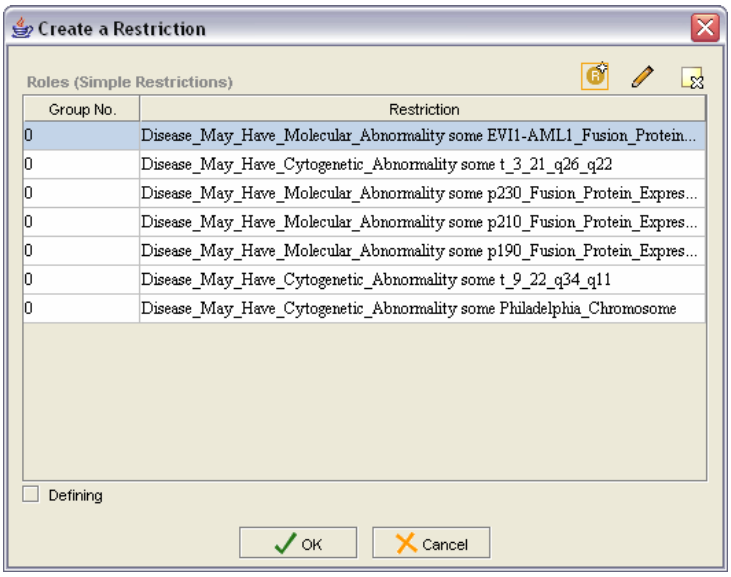


Figure 7.21 Multiple restrictions

12. 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.** (*Figure 7.22*).
2. Type a new number, then click elsewhere to deselect the field.
3. Repeat Steps *1.* and *2.* for each item in the group.
4. (Optional) If applicable, check the **Defining** box in the lower left of the window.

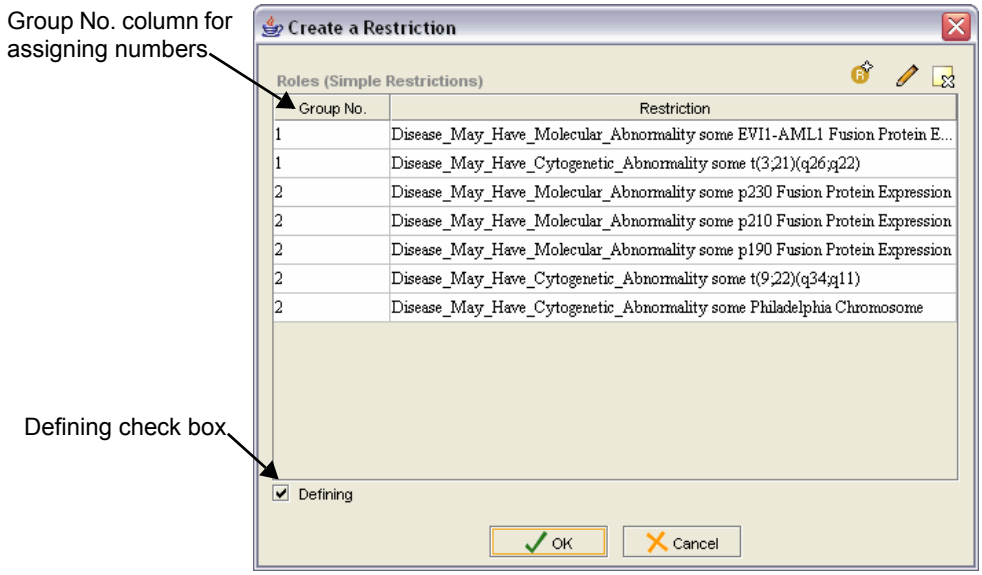


Figure 7.22 Role group example

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

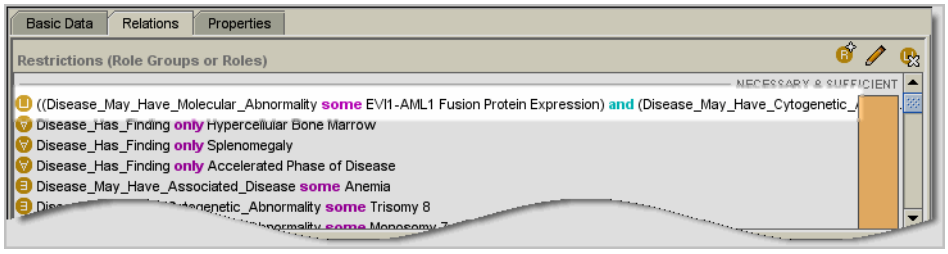


Figure 7.23 Role group expression

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

6. Click the **Save** button to save the change.
7. Click **OK** to close the save confirmation message window.

Adding an Association

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

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

To add an association, follow these steps:

1. Using the Class Browser, follow this path:

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

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

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

3. Click the **Relations** subtab.

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

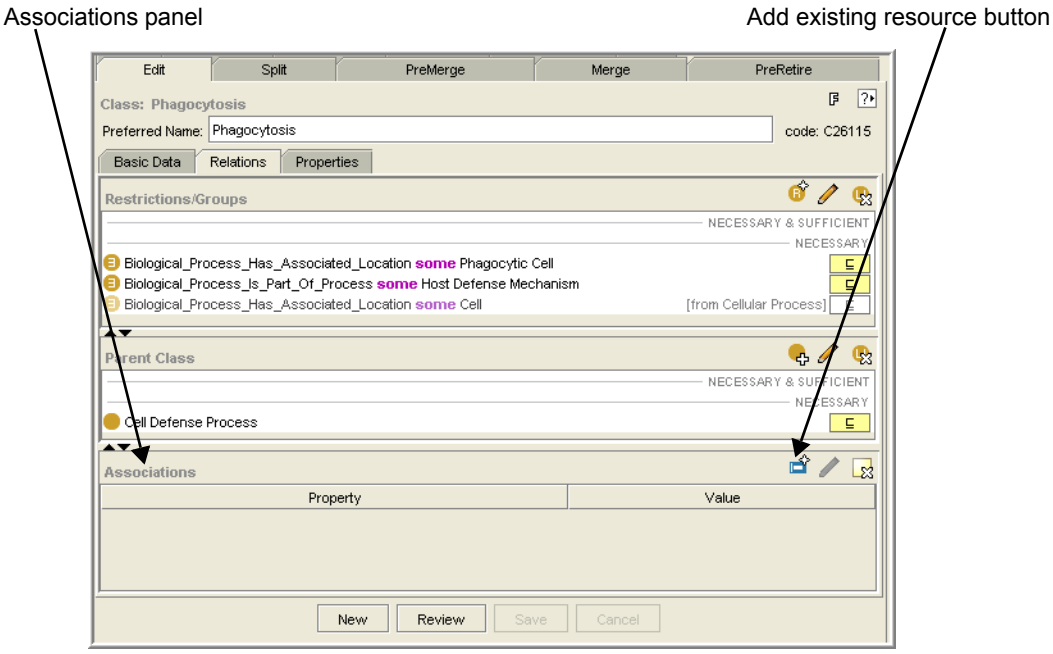


Figure 7.24 Relations subtab for Phagocytosis

The **Add an Object-Valued Property** window opens (*Figure 7.25* on page 118).

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

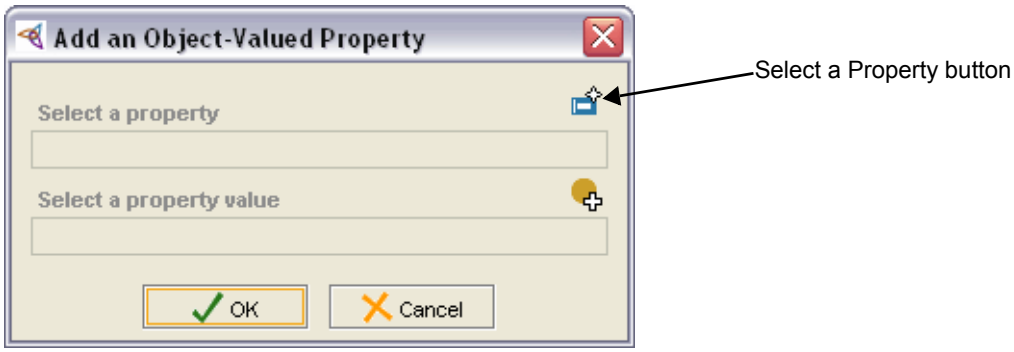


Figure 7.25 Add an Object-Valued Property window

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

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

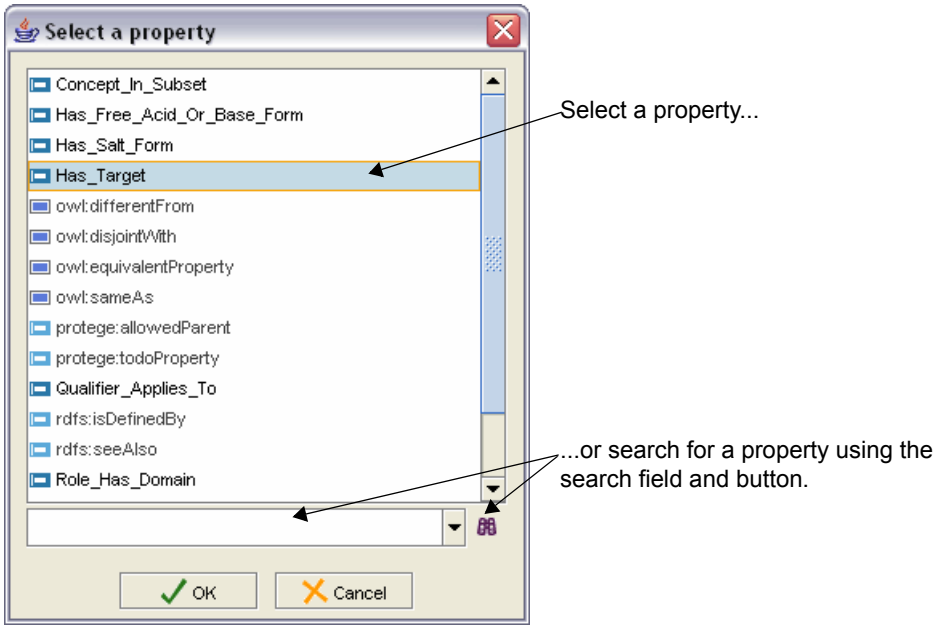


Figure 7.26 Select a Property window

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

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

In the **Add an Object-Valued Property** window, the selected value now appears in the **Select a property** field (*Figure 7.27* on page 119).

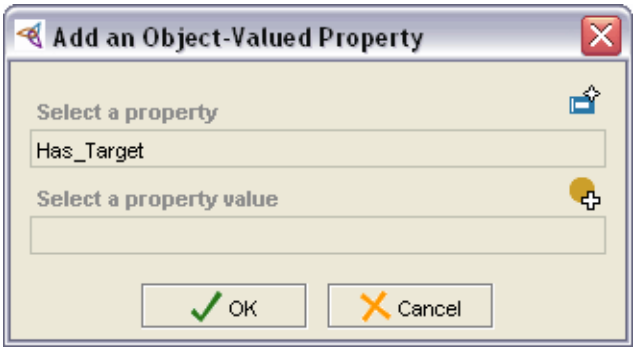


Figure 7.27 Select a property value added

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

   The **Select a property value** window opens.

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

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

10. Click **OK**.

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

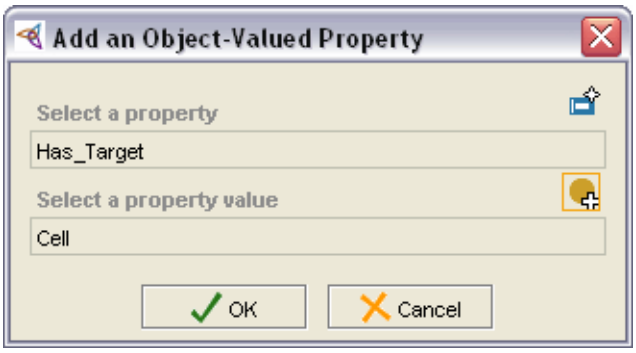


Figure 7.28 Add an Object-Valued Property window with stored values

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

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

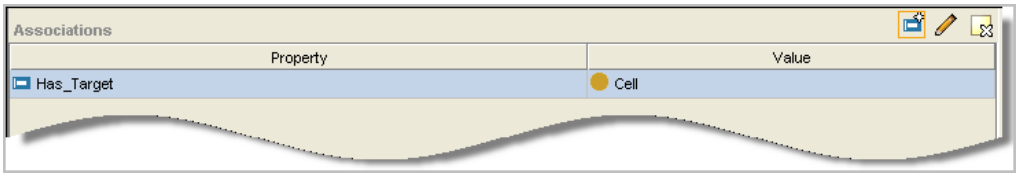


Figure 7.29 Associations panel

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

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

    A message confirms that the class has been saved successfully.

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

Editing Two Classes at the Same Time

Sometimes you may want to simultaneously edit two classes. For example, you may want to copy a property from one class to another.

Note: Although the **Copy** subtab is used for this task, copying (cloning) a class is a different procedure. For more information, see *Cloning a Class* on page 133.

To edit two classes simultaneously, follow these steps:

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

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

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

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

From Here...

- *Splitting, Merging, and Cloning Classes* on page 123
- *Pre-Retiring and Retiring Classes* on page 137
- *Using NCI Protégé Utilities* on page 147
- *Workflow Procedures for NCI Editors* on page 165

CHAPTER

8

SPLITTING, MERGING, AND CLONING CLASSES

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

Topics in this Chapter

- *Splitting a Class* on this page
- *Merging Classes* on page 127
- *Cloning a Class* on page 133

Splitting a Class

When to Consider Splitting a Class

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

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

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

About the Split Subtab

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

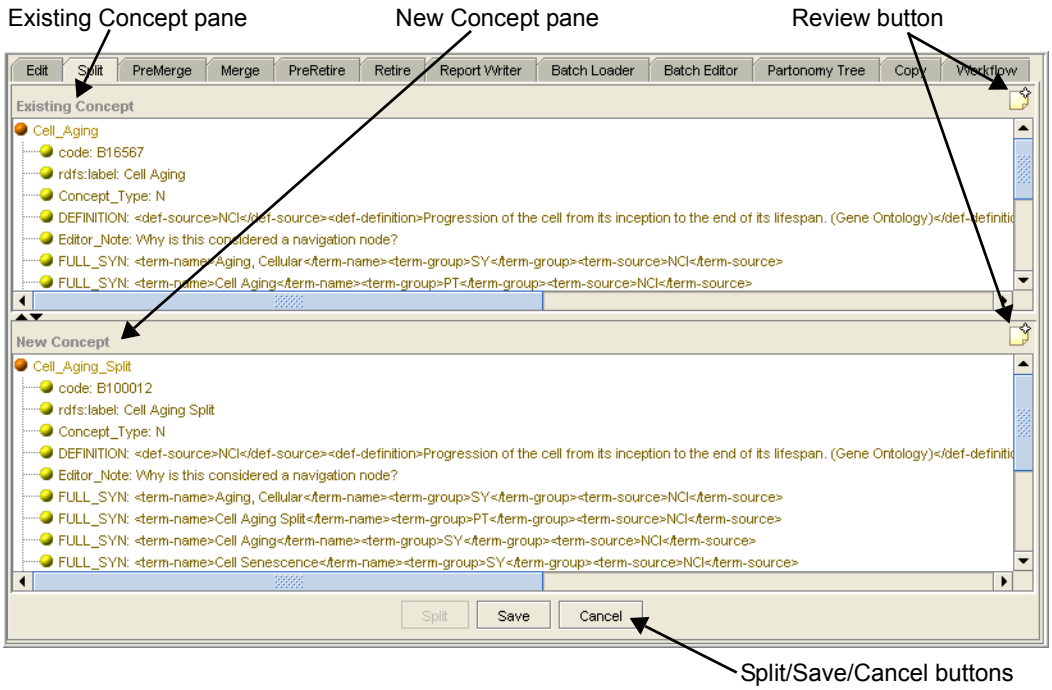


Figure 8.1 Split subtab

Table 8.1 describes the features of the **Split** subtab.

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

Table 8.1 Split subtab features

Using the Split Subtab to Split a Class

To split a class, follow these steps:

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

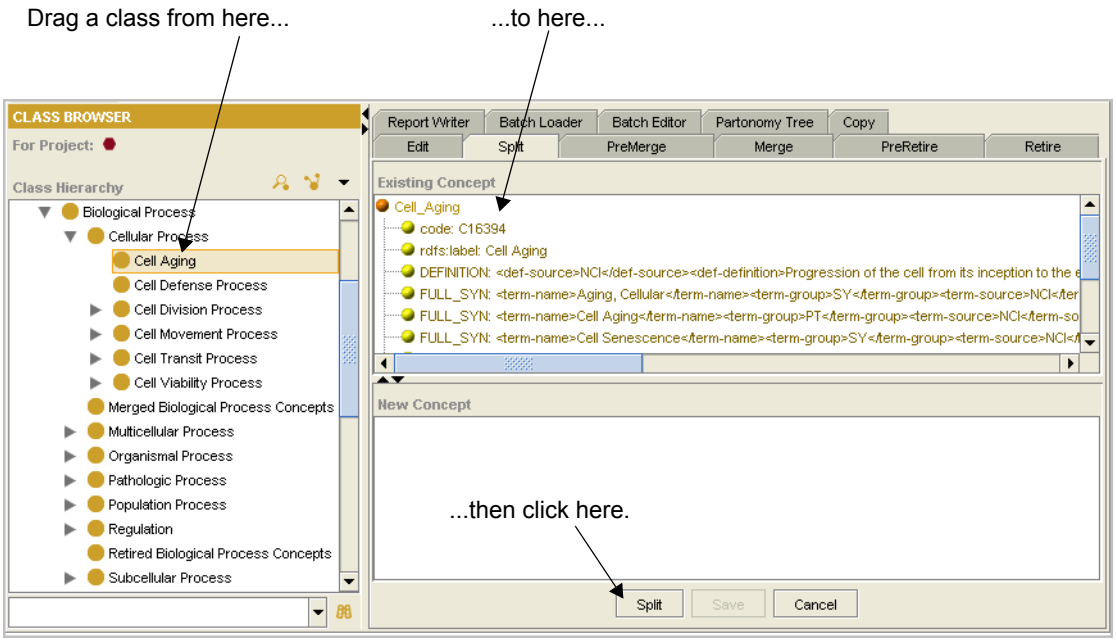


Figure 8.2 Existing Concept pane

The **Enter Class Identifiers** window opens.

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

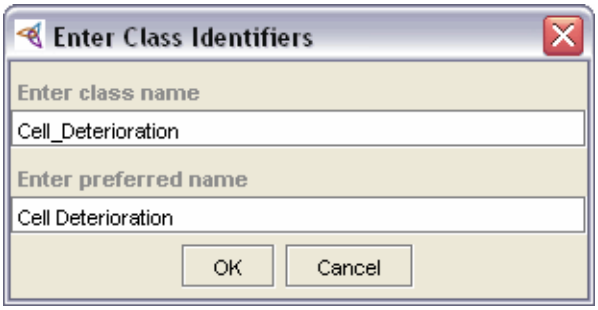


Figure 8.3 Enter Class Identifiers window

6. Click **OK**.

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

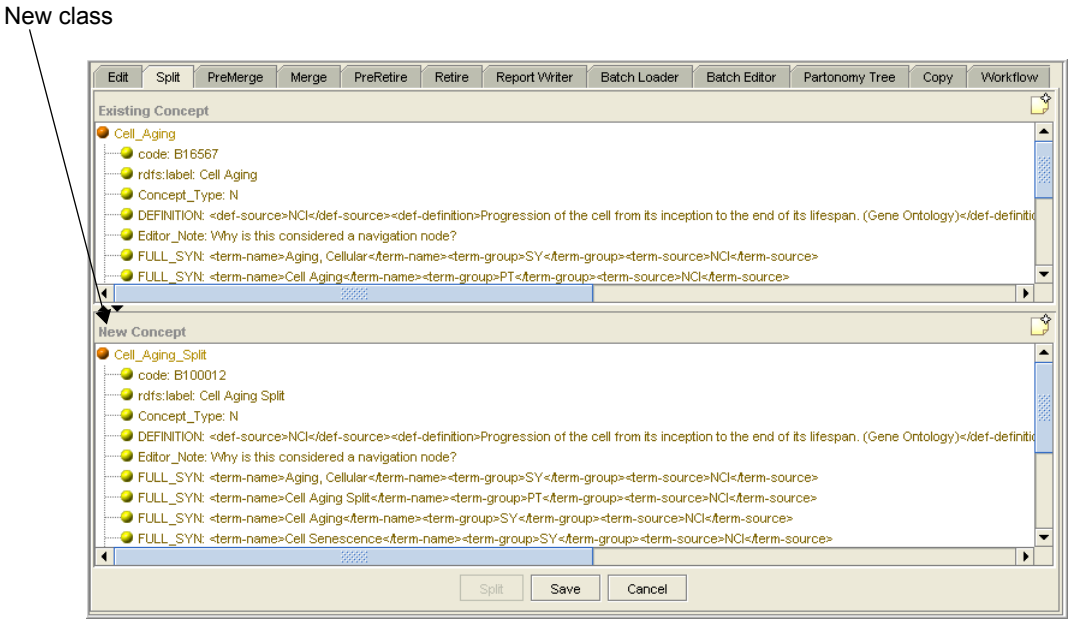


Figure 8.4 Split classes

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

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

Examining the Properties of a Split Class

1. Locate the new class in the Class Hierarchy.

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

2. Click the **Edit** subtab to view the properties and restrictions for the new class.
3. On the **Properties** subtab, note the *Split_From* property, which shows the code of the existing concept.

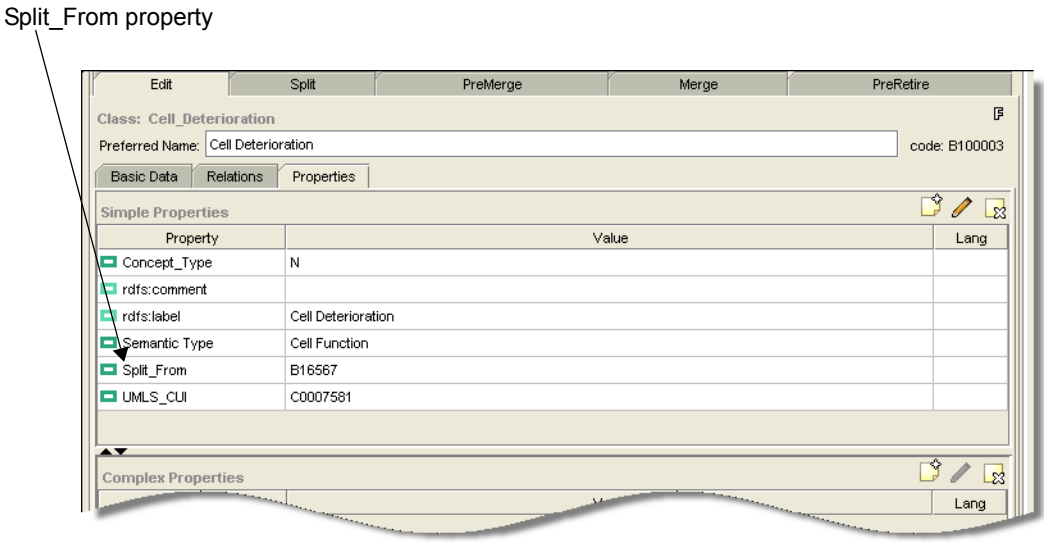


Figure 8.5 Properties of new class

Merging Classes

When to Consider Merging Classes

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

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

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

Merging is a two-step process:

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

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

About the PreMerge Subtab

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

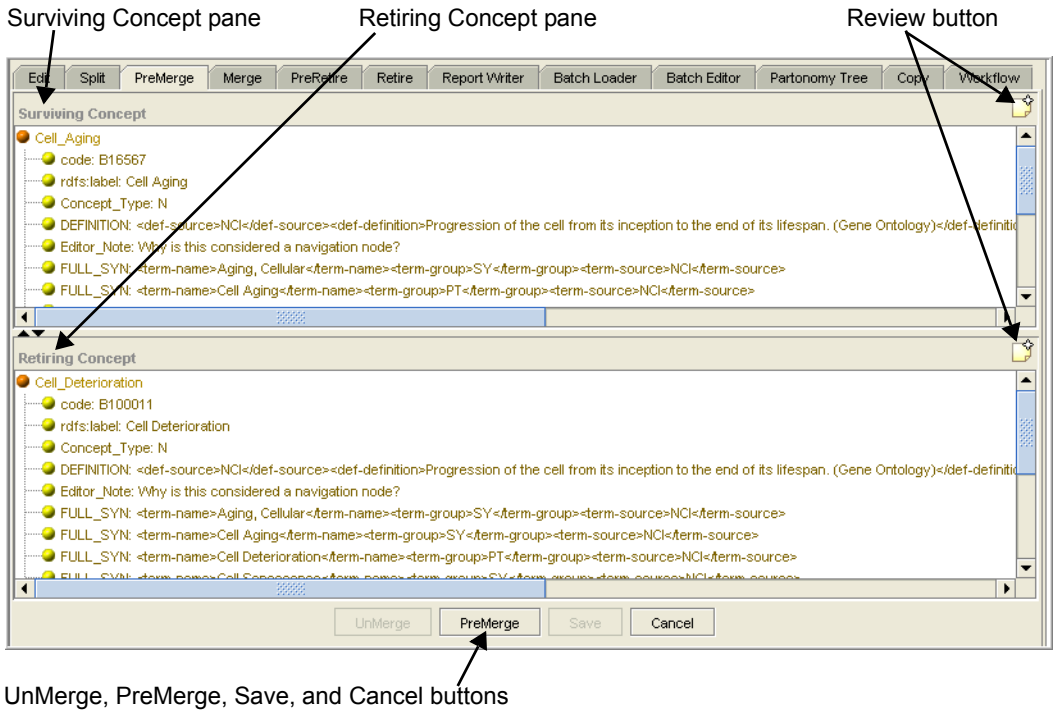


Figure 8.6 PreMerge subtab

Table 8.2 lists the features of the PreMerge subtab.

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

Table 8.2 PreMerge subtab features

Using the PreMerge Subtab to Flag Classes for a Merge

To pre-merge two classes, follow these steps:

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

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

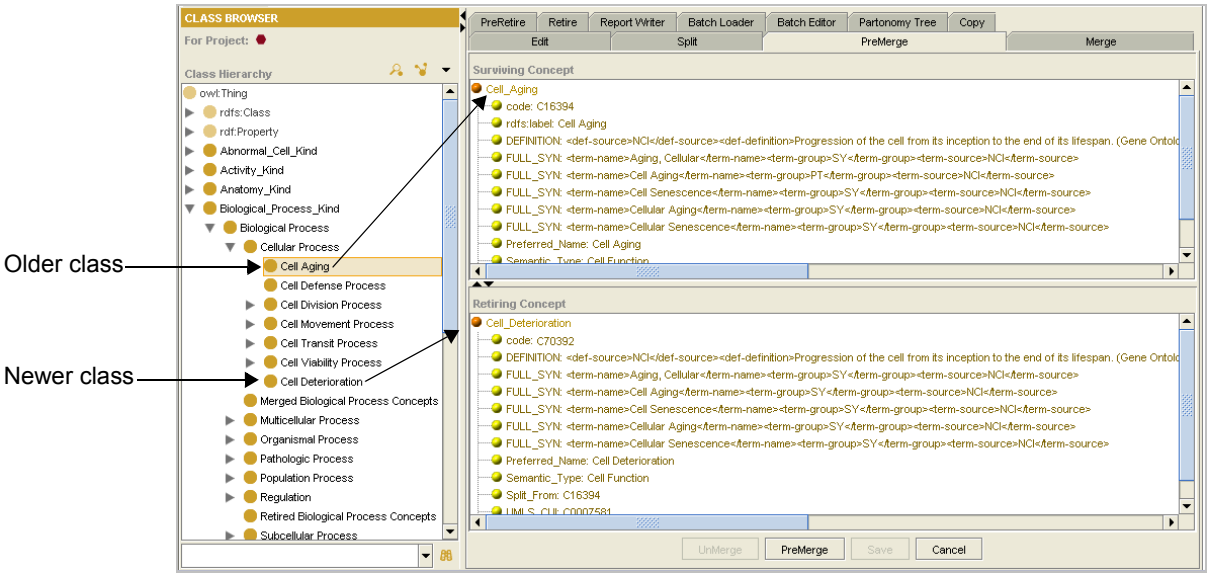


Figure 8.7 PreMerge subtab with surviving and retiring concepts

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

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

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

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

   A message confirms that the class has been saved successfully.

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

7. When you are ready to complete the pre-merge, click the **PreMerge** button at the bottom of the subtab.

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

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

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

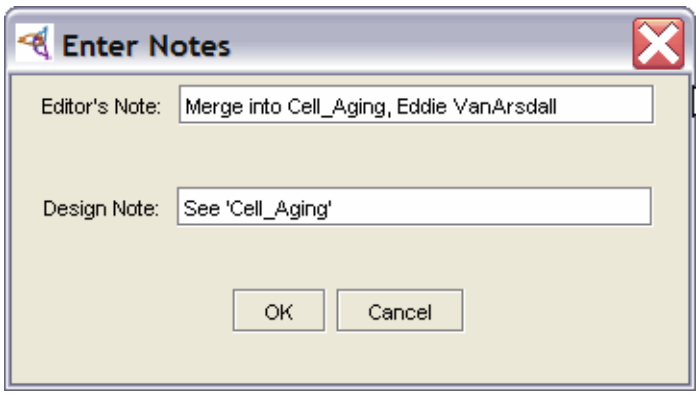


Figure 8.8 Enter Notes window

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

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

10. Click the **Save** button.

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

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

    The **Surviving Concept** and **Retiring Concept** panes are now empty.

Finding the Retiring Class under the Premerged_Concepts Branch

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

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

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

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

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

About the Merge Subtab

The **Merge** subtab enables workflow managers to merge two classes that have been flagged using the pre-merge action. For more information about the *pre-merge* action, see *Using the PreMerge Subtab to Flag Classes for a Merge* on page 129.

Figure 8.9 shows the layout of the Merge subtab.

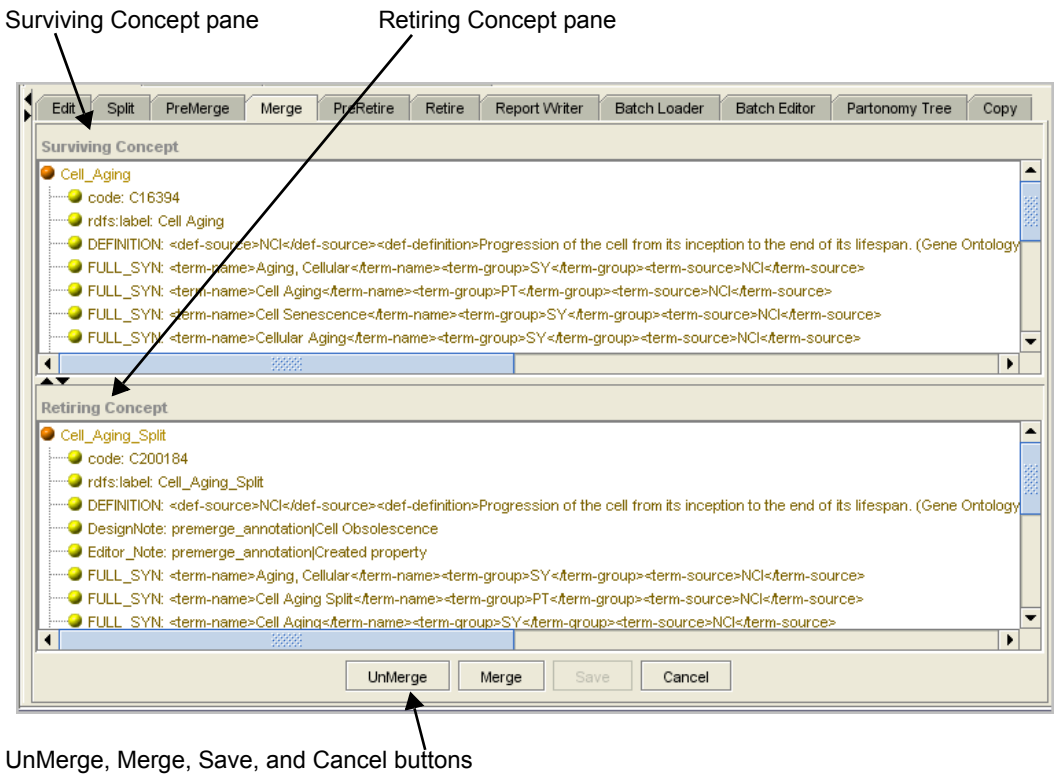


Figure 8.9 Merge subtab

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

Table 8.3 Merge subtab features

Using the Merge Subtab to Merge Flagged Classes

Note: Only workflow managers can perform this procedure.

To merge flagged concepts, follow these steps:

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

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

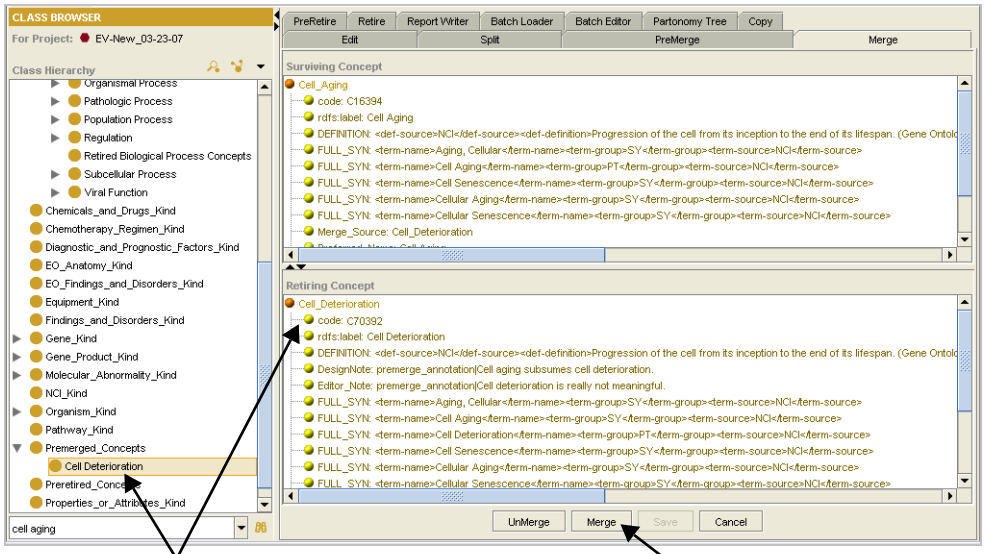


Figure 8.10 Merging two classes

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

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

   A message confirms that the two classes have been merged successfully.

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

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

Cloning a Class

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

To clone a class, follow these steps:

1. In the main Protégé window, click the **Copy** subtab.
2. Using the Class Browser, drag a class into the upper pane (*Figure 8.11*).

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

3. Click the **Clone** button.

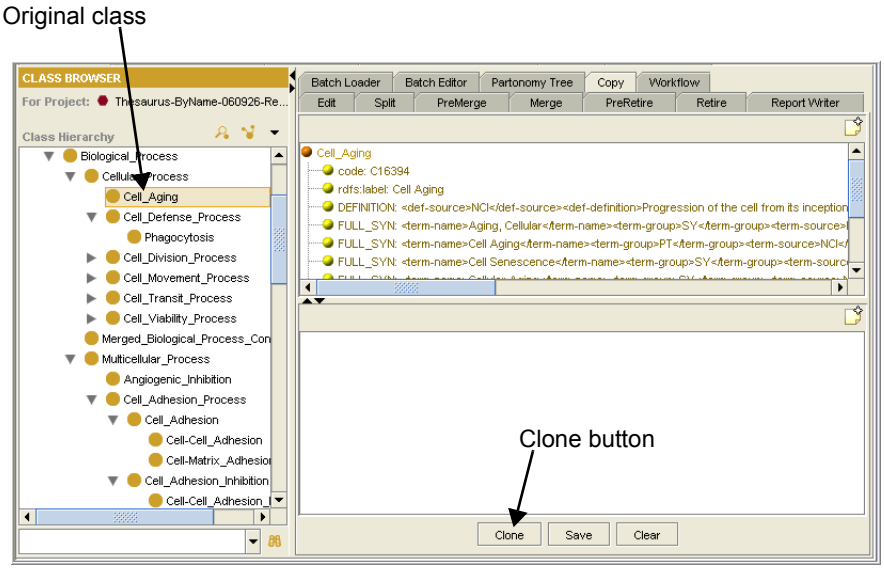


Figure 8.11 Copy subtab with original class

The **Enter Class Identifiers** window opens.

4. Enter the class name and preferred name (*Figure 8.12*).

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

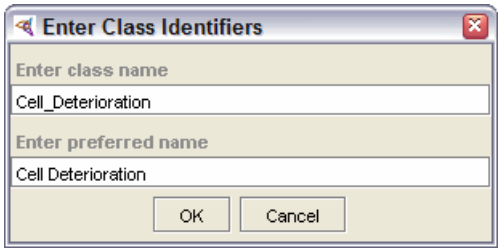


Figure 8.12 Enter Class Identifiers window

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

   The new class now appears in the lower pane of the **Copy** subtab (*Figure 8.13*).

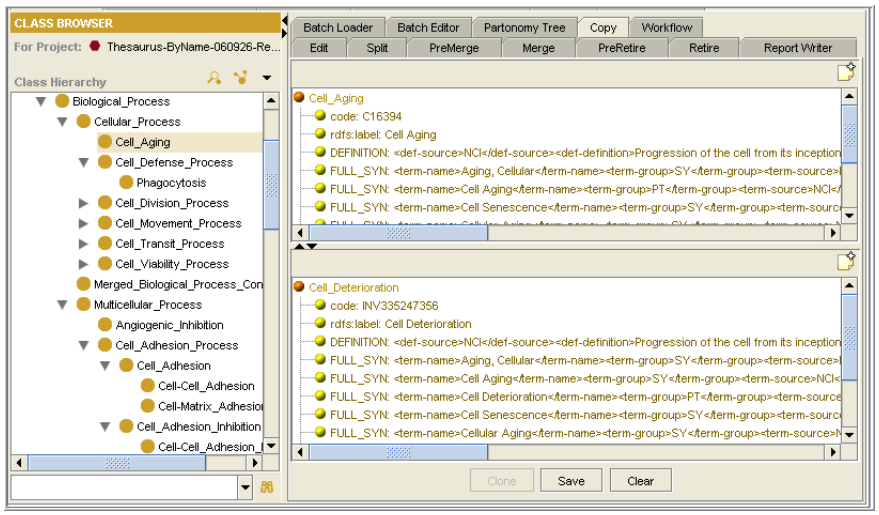


Figure 8.13 Copy subtab with newly created class

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

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

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

7. (Optional) Click the **Review** button in the upper or lower pane to view the state of either class before saving.
8. Click the **Save** button to save the changes.

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

From Here...

- *Pre-Retiring and Retiring Classes* on page 137
- *Using NCI Protégé Utilities* on page 147
- *Workflow Procedures for NCI Editors* on page 165

CHAPTER

9

PRE-RETIRING AND RETIRING CLASSES

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

Topic in this Chapter

- *About the Retirement Process* on this page
- *Using the PreRetire Subtab to Flag a Class for Retirement* on page 139
- *Retiring a Pre-Retired Class* on page 144

About the Retirement Process

When to Consider Retiring a Class

A class is typically retired when an editor determines that it is no longer needed. For example, another better-modeled class may have the same meaning.

PreRetire vs. Retire

Retiring is a two-step process:

1. An *editor* (modeler) applies a *PreRetire* flag, which designates a class for retirement.

   The flagged class is moved to a **Preretired_Concepts** branch in the Class Hierarchy.

2. During a baseline review and update, the *workflow manager* reviews the pre-retired candidates and either accepts or rejects them.

   Retired classes are deprecated but remain accessible in search results. Their names are preceded by a letter D enclosed in square brackets: [D].

Pre-Retiring a Class

About the PreRetire Subtab

The **PreRetire** subtab enables you to flag a class for retirement. This subtab includes two subtabs that show, respectively, subclasses and referencing classes.

Figure 9.1 shows the layout of the **PreRetire** subtab.

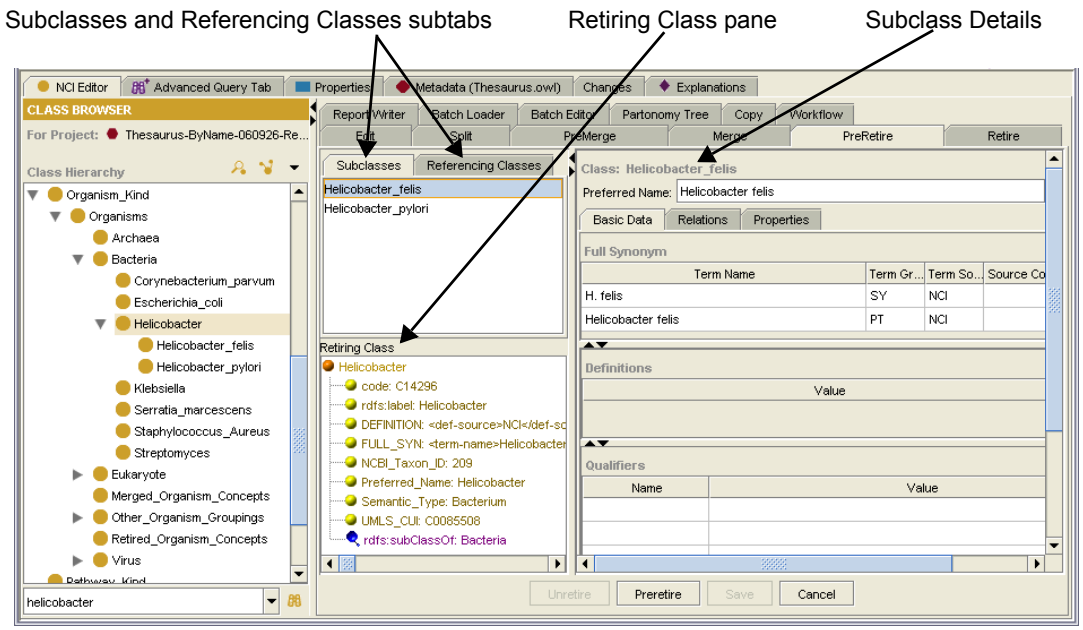


Figure 9.1 PreRetire subtab

Table 9.1 describes the features of the PreRetire subtab.

Feature	***Description***
Subclasses subtab	Shows the names of all subclasses of the currently selected class.
Referencing Classes subtab	Shows the names of all subclasses that have the currently selected class as a referred class.
Basic Data, **Relations**, and **Properties** subtabs	Show basic data, relations, and properties for a selected subclass or referencing class.
Retiring Class pane	Shows a tree representation of the selected class.
Unretire button	Reverses a pre-retire action.
Preretire button	Flags a selected class for retirement.
Save button	Saves a selected class with a pre-retire flag.
Cancel button	Discards unsaved changes.

Table 9.1 PreRetire subtab features

Using the PreRetire Subtab to Flag a Class for Retirement

You can designate a class for retirement by applying a pre-retire flag. The pre-retire procedure requires the following primary steps:

- *Selecting a Class for Retirement* on this page
- *Eliminating Subclass Dependencies* on this page
- *Saving the Class with a Pre-Retired Status* on page 143
- *Finding the Pre-Retired Class in the Hierarchy* on page 143

Selecting a Class for Retirement

To select the class that will be retired, follow these steps:

1. In the main Protégé window, click the **PreRetire** subtab.
2. Using the Class Browser, drag a class into the **Retiring Class** panel on the lower left side of the **PreRetire** subtab (*Figure 9.1* on page 138).

   If subclasses exist for the selected class, they appear in the **Subclasses** panel on the upper left. If other classes reference the selected class by role relations, you can see them by clicking the **Referencing Classes** subtab.

Eliminating Subclass Dependencies

When flagging a class for retirement, you first eliminate any dependencies for the class by assigning each of its subclasses to another parent class.

To eliminate subclass dependencies, follow these steps for each subclass:

1. Expand the **PreRetire** subtab by clicking on the **Expand** button on the top right (*Figure 9.2*).

   This hides the Class Browser and helps you to see all of the details on the middle and right panels.

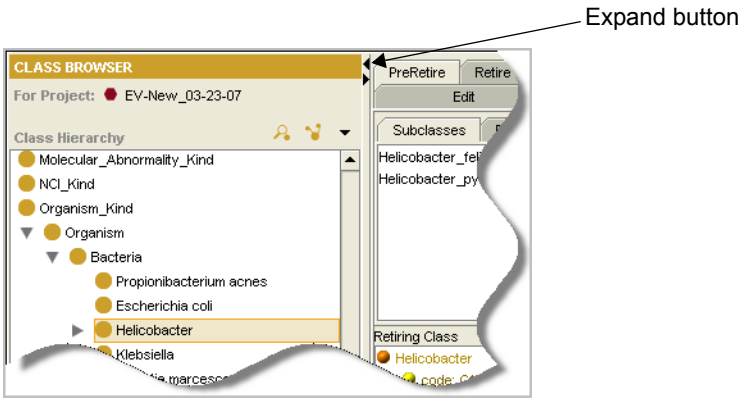


Figure 9.2 Hiding the Class Browser

2. Select a subclass in the **Subclasses** panel (upper left).

Detailed information for the selected subclass now appears in the Class Editor pane on the right (*Figure 9.3*).

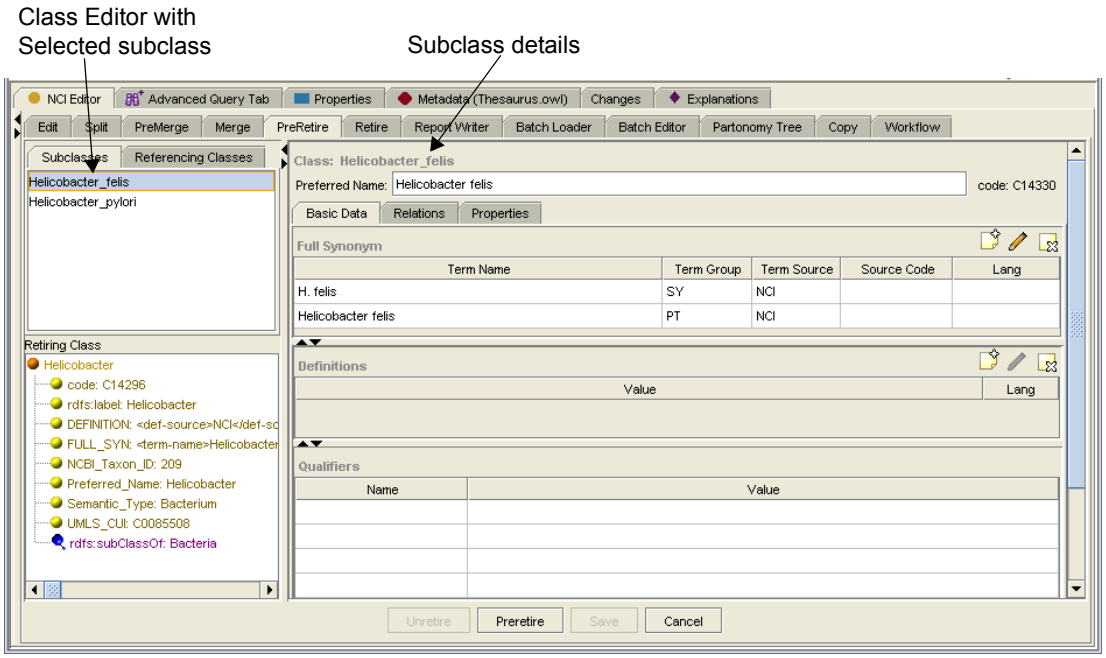


Figure 9.3 Selected subclass with Basic Data subtab displayed

3. Click the **Relations** subtab to view parent classes for the selected class.

   The current example has only one parent class: *Helicobacter* (*Figure 9.4*). When only one named parent class exists, you cannot delete it.

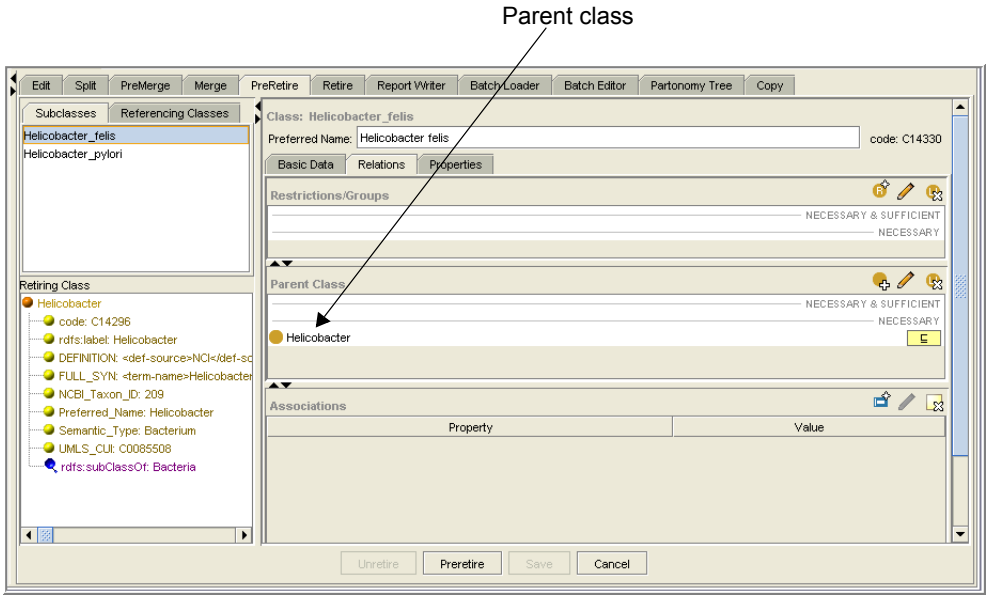


Figure 9.4 Relations subtab with one parent class

To retree the subclass *Helicobacter_felis*, you first need to add a second superclass, as in the following steps.

4. Click the **Add parent class** button in the upper right area of the **Parent Class** panel.

   The **Select a class** window opens (*Figure 9.5*).

5. Select or search for a class.

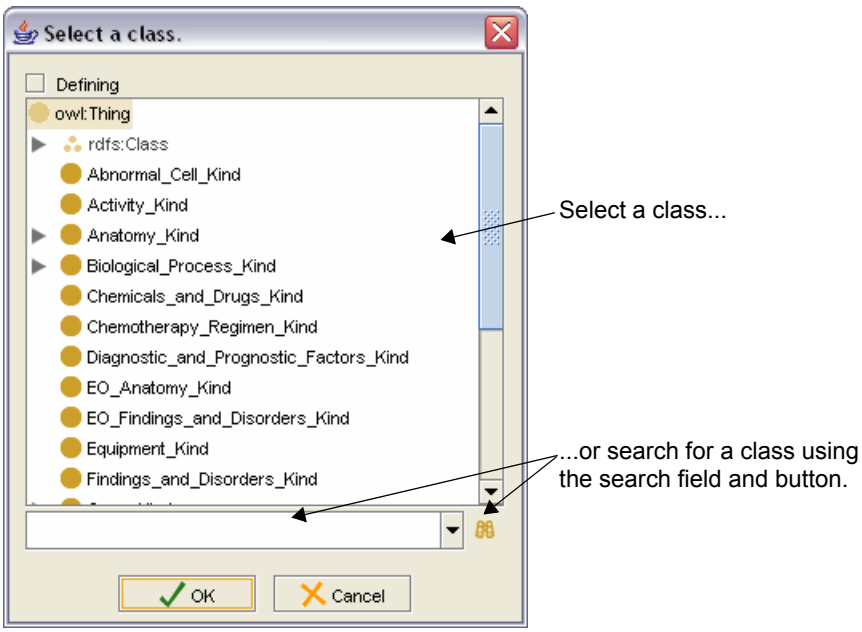


Figure 9.5 Select a class window

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

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

   In the current example, there are now two parent classes: *Bacteria* and *Helicobacter*.

7. Select the class to be deleted.

   For the current example, select the **Helicobacter** class.

8. Click the **Delete selected row** button in the upper right area of the **Parent Class** panel.

9. Click **Yes** to confirm deletion of the selected class.

Note: The **Basic Data**, **Relations**, and **Properties** subtabs are nested inside of the **PreRetire** subtab, so you may not notice that there are two separate sets of buttons at the bottom (*Figure 9.6*).

In the next step, make sure that you click the **Save** button at the bottom of the *inner* subtab to re-tree each subclass. When you are ready to flag the class for retirement, click the **PreRetire** button at the bottom of the *outer* subtab. You will then be asked to save the change using the *outer* **Save** button.

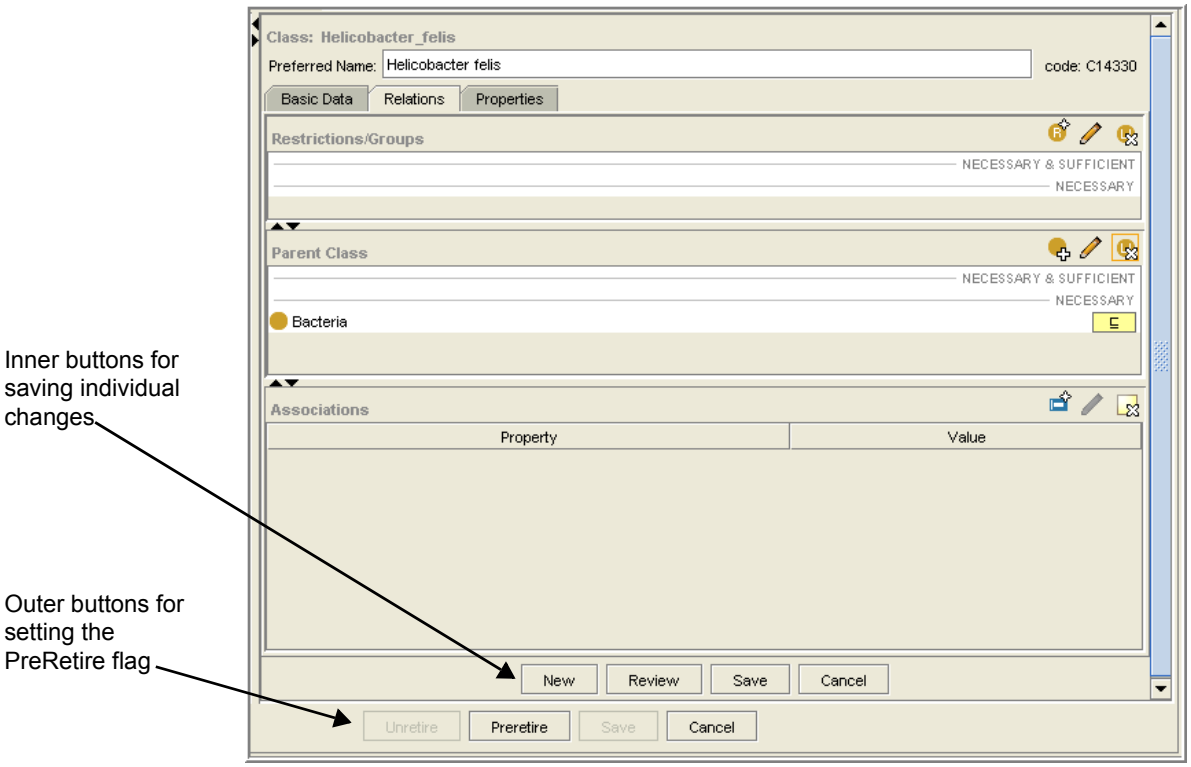


Figure 9.6 Buttons sets for inner, nested subtabs and outer subtab

10. Click the *inner* **Save** button to accept the change.

    The selected subclass disappears from the **Subclasses** list.

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

12. Repeat steps *2.* through *11.* for all classes shown in the **Subclasses** panel.

    When you are finished, there should be no subclasses shown.

Saving the Class with a Pre-Retired Status

To pre-retire the class and save the change, follow these steps:

1. Click the **PreRetire** button at the bottom of the **PreRetire** subtab.

   The **Enter Notes** window opens and displays the following fields (*Figure 9.7*):

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

2. Modify the **Editor's Note** and **Design Note** if necessary.

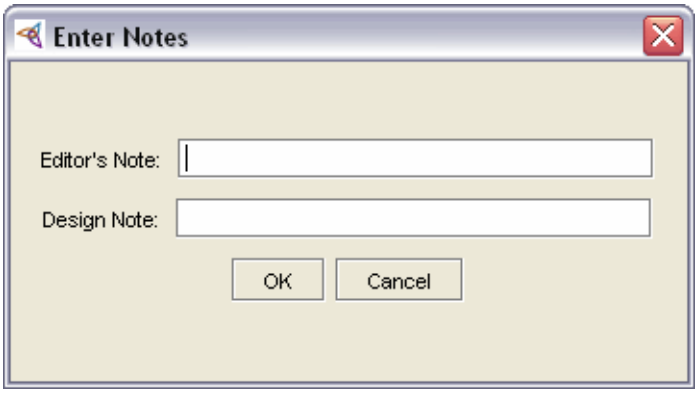


Figure 9.7 Enter Notes window

3. Click **OK** to close the **Enter Notes** window.
4. Click the *outer* **Save** button to accept the change.

   A message confirms that the selected class has been flagged for retirement.

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

Finding the Pre-Retired Class in the Hierarchy

To locate the pre-retired class, follow these steps:

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

   The **Preretired_Concepts** branch displays classes that are retirement.

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

   Among other properties, the **Simple Properties** panel shows the following information:

   - Your design note
   - Your editor's note
   - The re-treed subclasses and their parent

Retiring a Pre-Retired Class

Note: Only workflow managers can perform this procedure.

During a baseline review and update, workflow managers review the candidates designated for retirement and either accept or reject them. The procedures in this section explain how to accept a class for retirement.

About the Retire Subtab

The **Retire** subtab enables authorized users (*e.g.*, workflow managers) to retire a class that has been flagged by a pre-retire action. This subtab has a single pane that shows a tree representation of the selected class.

Figure 9.8 shows the layout of the **Retire** subtab.

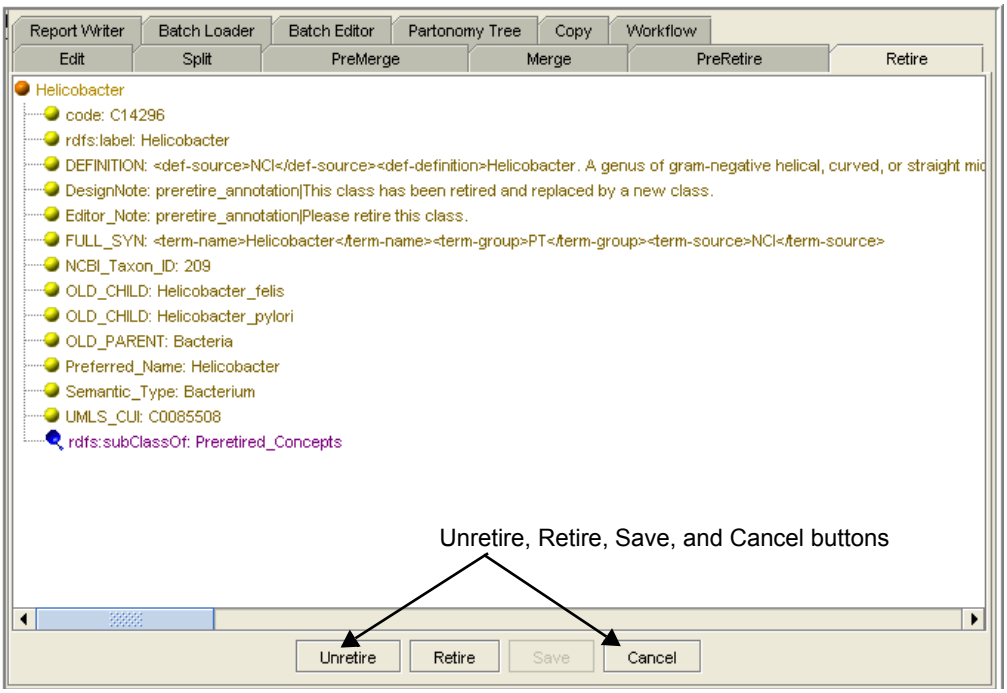


Figure 9.8 Retire subtab

Table 9.2 lists the features of the Retire subtab.

Feature	***Description***
Unretire button	Removes a previously set pre-retire flag.
Retire button	Retires a class.
Save button	Saves changes and formally retires the class by re-treeing it from *pre-retired* to *retired*.
Cancel button	Discards unsaved changes.

Table 9.2 Retire subtab features

Using the Merge Subtab to Retire a Flagged Class

To retire a concept, follow these steps:

1. Using the Class Browser, locate the **Pretired_Concepts** branch.
2. Expand the branch, then select the class to be retired.
3. Click the **Retire** subtab.
4. Drag the selected class onto the **Retire** subtab.
5. Click the **Retire** button at the bottom of the subtab.
6. Click the **Save** button.

   A message confirms that the class has been retired.
7. Click **OK** to close the message window.

   The **Retire** subtab is now empty.

From Here...

- *Using NCI Protégé Utilities* on page 147
- *Workflow Procedures for NCI Editors* on page 165

CHAPTER

10

USING NCI PROTÉGÉ UTILITIES

This chapter introduces three useful NCI Protégé utilities: the **Report Writer**, **Batch Loader**, and **Batch Editor** subtabs.

Topics in this Chapter

- *Generating Reports for Selected Classes* on this page
- *Loading a Batch of Classes for Editing* on page 154
- *Loading a Batch of Edited Classes* on page 159

Generating Reports for Selected Classes

Protégé enables you to generate a report for a selected class. You can choose to have the report show only the parent class and its subclasses, or you can choose to also show the properties and restrictions for the class.

About the Report Writer Subtab

Use the **Report Writer** subtab to produce a report for a selected class. *Figure 10.1* (page 148) shows the simple **Report Writer** subtab interface, which is used mainly to open the **Report Writer** window.

In the window you can specify the name and location of an external report file (usually a text file). You can then view the report in a text editor such as Notepad.

Caution: When using the Report Writer, note that if you want to run a report on a high-level class in the tree (such as pharmacological agents), you may experience an extended wait time while the report runs. This could prevent you from performing other 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.

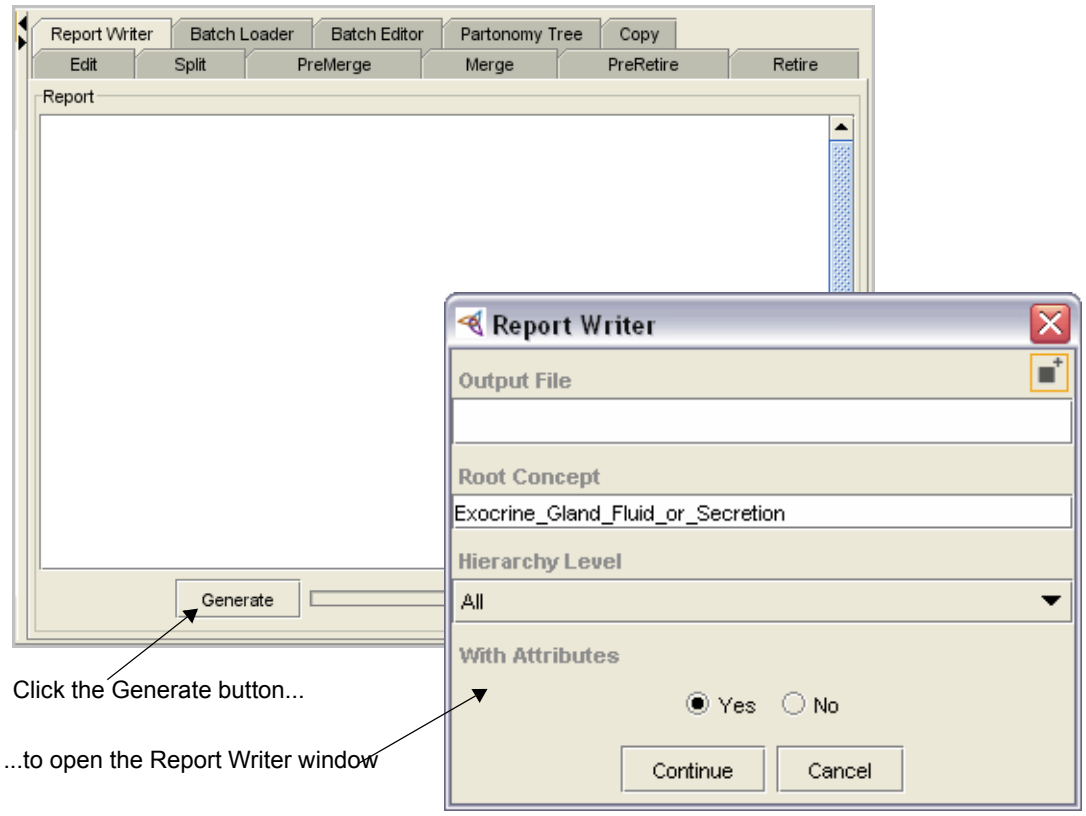


Figure 10.1 Report Writer subtab and Report Writer window

Using the Report Writer Subtab to Generate Reports

Creating an Output Directory and Files

Each report that you generate requires a specified output directory and file. For best results, create a local or network directory for storing report files.

Caution: If you do not specify a directory, Protégé stores report files in its own application directory (*C:\Program Files\Protege.Client-1.2.x*). If you store report files in the same directory as application files, you risk inadvertently deleting files that could cause Protégé not to run properly or not to run at all.

While naming report files, consider their purpose. For example, if you need a report for the class *Respiratory System Fluid or Secretion*, you can create one file that shows only the parent class and its children, and another file that shows parents, children, properties, and restrictions. Append the *.txt* extension to each file name, as in the following examples:

- *respiratory_system_parent-child-only.txt*
- *respiratory_system_all.txt*

Generating a Report

To generate a report for a selected class, follow these steps:

1. Using the Class Browser, select the class that will be the subject of the report.

   Note: This is a required step for generating a report. If you select the root of the hierarchy (*owl:Thing*), the Report Writer will prompt you to select another root.

2. Click the **Report Writer** subtab.

   As shown in *Figure 10.2*, the **Report Writer** subtab shows mostly white space. It does not display report output. Its main purpose is to generate an external text file that you can open and view in a text editor such as Notepad.

3. With the class still selected in the Class Browser on the left, click the **Generate** button (shown in *Figure 10.2*).

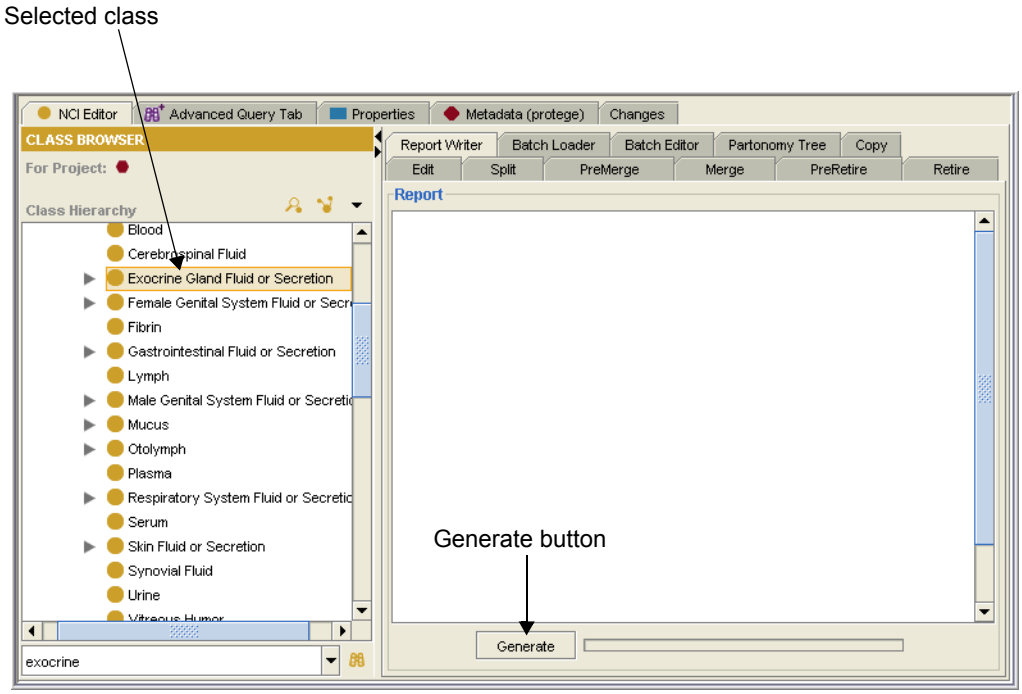


Figure 10.2 Report Writer subtab - Generate button

The **Report Writer** window opens. The selected class name is displayed in the **Root Concept** field (*Figure 10.3* on page 150).

Note: Although you can enter a new root concept name, the Report Writer will still generate a report for the class that you selected in the first step. Typing a new value will not change the result.

4. Select the output file for the report by following these steps:

   a. Click the **Browse for File** button in the upper right area of the window.

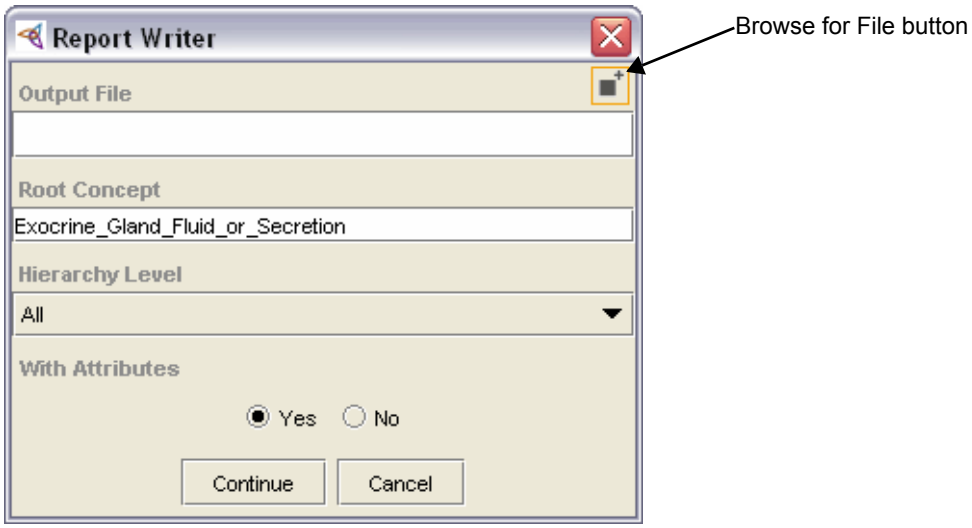


Figure 10.3 Report Writer window for specifying an output file

   b. In the **File Browser** window (shown in *Figure 10.4*), do either of the following:

      — If you are creating a *new* output file, browse to the location where the file should be stored, then type a name in the **File Name** field. Make sure that you include the `.txt` extension.

      — If you are using an existing output file, browse to the appropriate directory, then select the file.

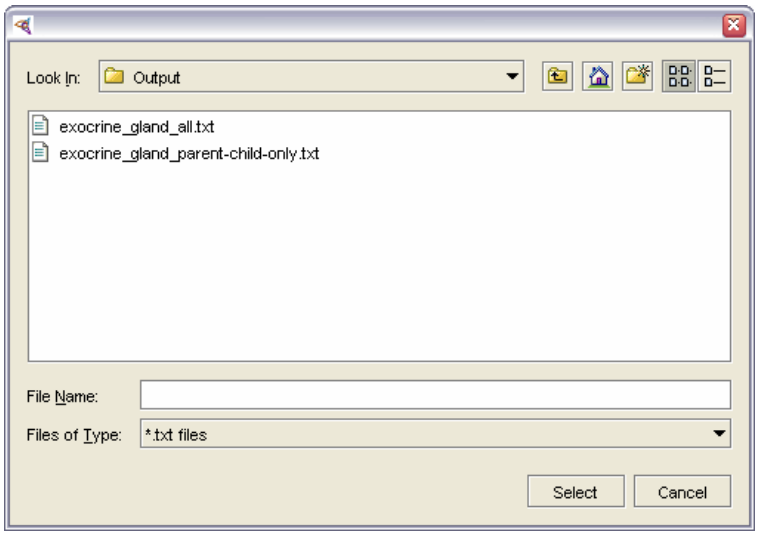


Figure 10.4 File browser window

   c. Click the **Select** button to close the file browser window.

      Back in the **Report Writer** window, the file path and name now appear in the **Output File** field.

5. Specify the levels of the hierarchy to be included in the report by selecting a value from the hierarchy list.

   The **All** level and the first three numerical levels display the following information:

   - **All** includes the parent (root) class and all subclasses.
   - **Level zero (0)** displays only the parent class.
   - **Level one (1)** displays the parent class and its subclasses (children).
   - **Level two (2)** displays the parent class, its subclasses, and any children of the subclasses.

   The results of selecting the remaining levels vary according to the number of children that the selected parent has. *Figure 10.5* shows the level numbers of the current example, *Exocrine Gland Fluid or Secretion*.

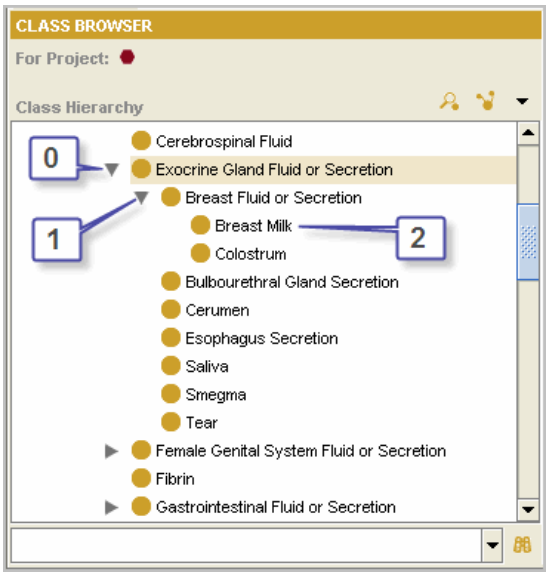


Figure 10.5 Level numbers for Exocrine Gland Fluid or Secretion

6. Select one of the options under the **With Attributes** section:

   - Select **Yes** to generate a report that includes the parent, all subclasses, and all properties and restrictions; or
   - Select **No** to include only the parent and all subclasses.

To show a simple report first, the example in *Figure 10.6* uses the **No** option.

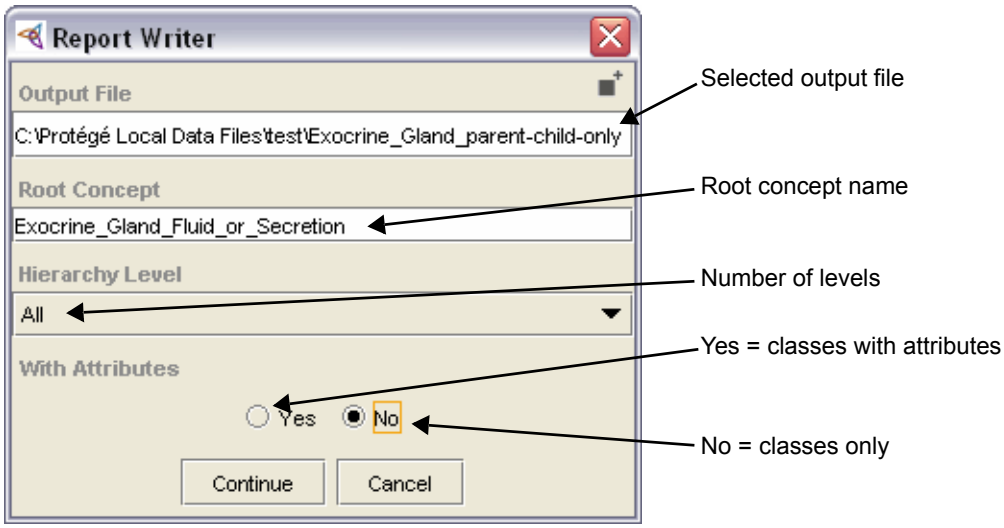


Figure 10.6 Report Writer window with completed values

7. Click the **Continue** button.

   The **Report Writer** status window appears in the upper left corner of your computer screen (*Figure 10.7*).

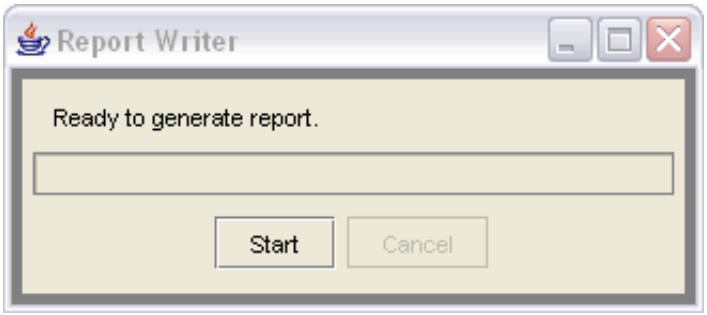


Figure 10.7 Report Writer status window as it initially appears

8. Click the **Start** button.

   The window displays a progress indicator bar.

9. When you see a *Report generation completed* message, click the standard Windows **Close** button in the upper right corner to close the window.

   Caution: Protégé opens a status window for each report that you generate, so if you generate ten reports, you will open ten windows. Remember to close the status window after each report is generated.

10. Using Windows Explorer, browse to the output file that you used for the report.

11. Open the file in a text editor such as Notepad.

    The file should resemble the sample report shown in *Figure 10.8*.

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.

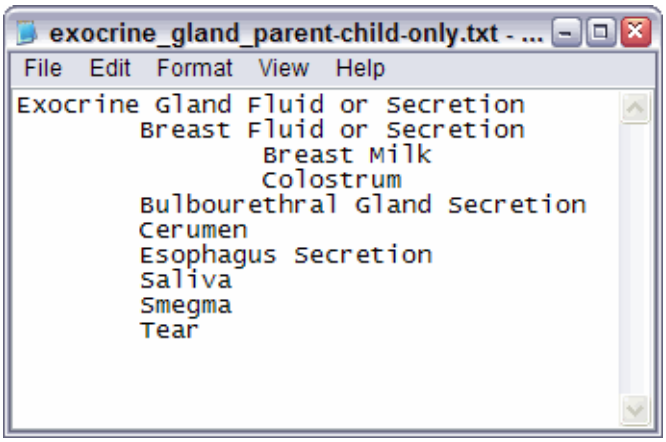
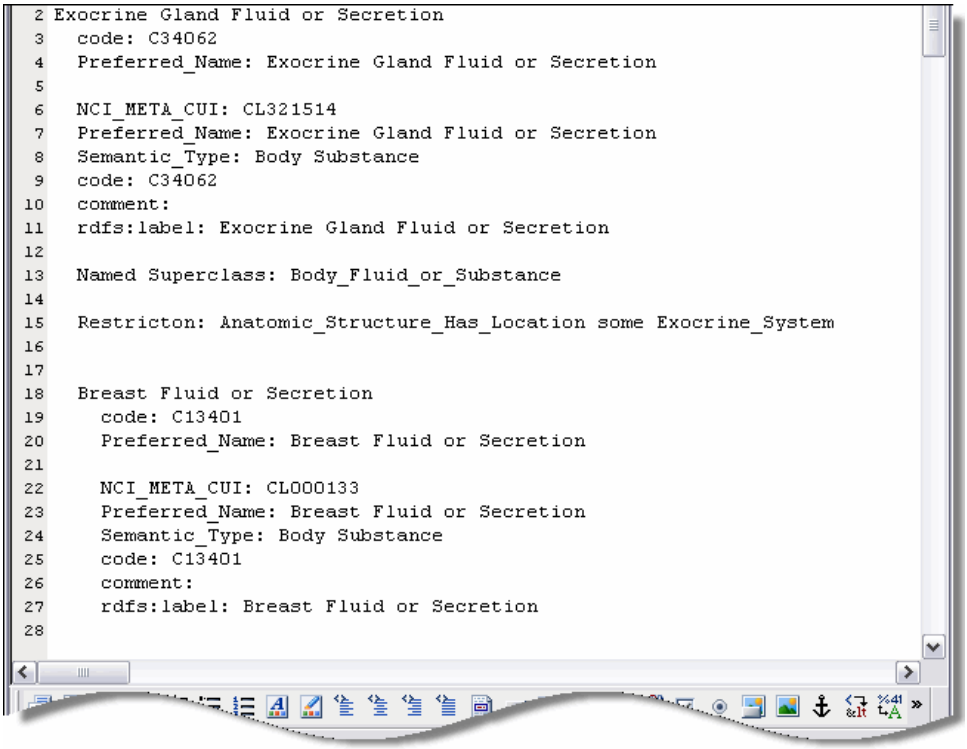


Figure 10.8 Report with classes only - no attributes

The sample report shown in *Figure 10.9* was created by selecting the **Yes** option. This report shows classes, subclasses, properties, and restrictions.



```
 2 Exocrine Gland Fluid or Secretion
 3   code: C34062
 4   Preferred_Name: Exocrine Gland Fluid or Secretion
 5
 6   NCI_META_CUI: CL321514
 7   Preferred_Name: Exocrine Gland Fluid or Secretion
 8   Semantic_Type: Body Substance
 9   code: C34062
10   comment:
11   rdfs:label: Exocrine Gland Fluid or Secretion
12
13   Named Superclass: Body_Fluid_or_Substance
14
15   Restricton: Anatomic_Structure_Has_Location some Exocrine_System
16
17
18   Breast Fluid or Secretion
19     code: C13401
20     Preferred_Name: Breast Fluid or Secretion
21
22     NCI_META_CUI: CL000133
23     Preferred_Name: Breast Fluid or Secretion
24     Semantic_Type: Body Substance
25     code: C13401
26     comment:
27     rdfs:label: Breast Fluid or Secretion
28
```

Figure 10.9 Report with classes, subclasses, properties, and restrictions

12. Try generating several reports with and without attributes.

Loading a Batch of Classes for Editing

Protégé enables you to load a batch of classes for editing. This is useful when you want to import a large number of classes and then edit them in the Protégé environment.

About the Input File Format for a Batch Load

You first create the batch file by specifying the following basic data for each class. Each of the three fields is case-sensitive:

- **Field 1:** The name of the class (with underscores)
- **Field 2:** The preferred name of the class
- **Field 3:** The name of the parent class.

The batch-loading process accepts a tab-delimited (text) input file using the specified format. You can set up input file in a text editor such as Notepad or in a spreadsheet program such as Excel. In either case, save the file as a tab-delimited text file (`.txt`).

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

About the Batch Loader Subtab

The **Batch Loader** subtab (*Figure 10.10*) manages the process of importing a batch of classes. The Log area is empty until you specify and load an input file.

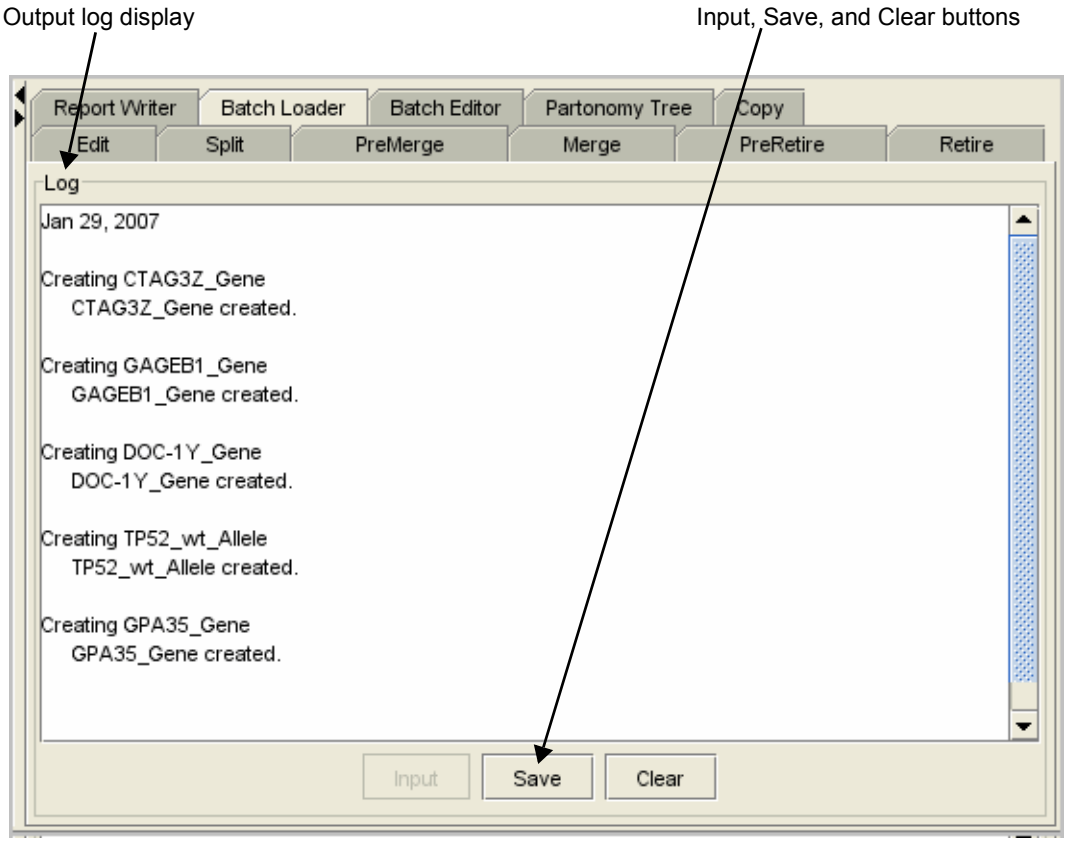


Figure 10.10 Batch Loader subtab

Table 10.1 lists the features of the Batch Loader subtab.

Feature	***Description***
Output log display	Shows the log status (the status of the batch load). If the load is successful, the contents of the output log show here. If the input file contains errors, such as entries that are already in the knowledge base, an error message shows here. If you need to start over, click the **Clear** button.
Input button	Opens a window in which you can specify an input file and an output log file.
Save button	Saves the log to an ASCII file and confirms the file creation.
Clear button	Clears the display area.

Table 10.1 Batch Loader subtab features

Using the Batch Loader Subtab to Load an Input File

To load a batch of classes, follow these steps:

1. Using a text editor such as Notepad, prepare and save a tab-delimited input file such as the example shown in *Figure 10.11*.

   Make sure that the file contains the class name, preferred name, and name of the parent class, as specified on page 154.

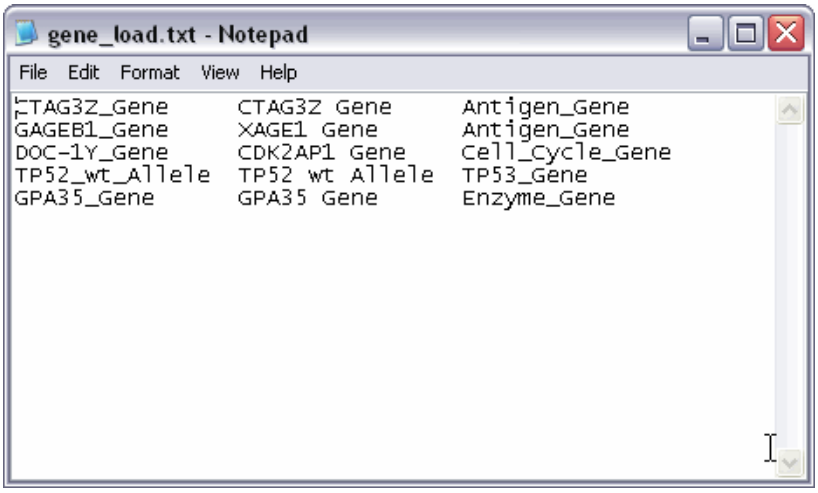


Figure 10.11 Tab-delimited input file for batch load

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

   Note: In this step, the class you select has no bearing on the batch load procedure. You are performing this step because you cannot use the Batch Loader subtab unless something other than *owl:Thing* is selected in the Class Browser.

3. Click the **Batch Loader** subtab.

The subtab is initially blank, as shown in *Figure 10.12*. Once you set up and run the batch load process, the display area of the subtab shows the output of your batch load file.

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

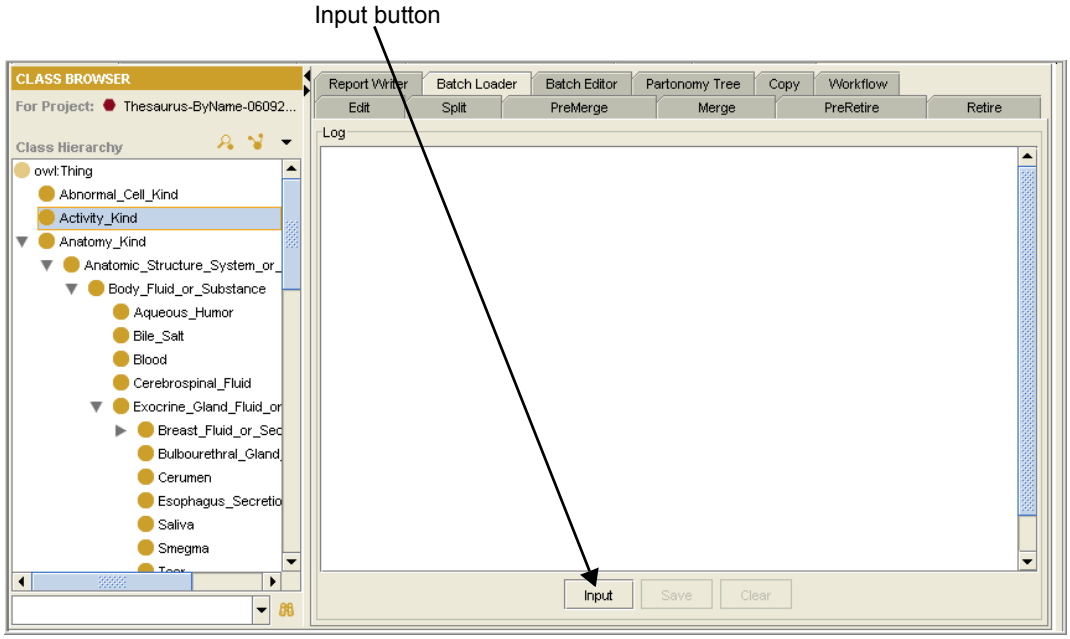


Figure 10.12 Batch Loader subtab

The **Batch Loader** window opens (*Figure 10.13*).

5. Browse for input and output log files by following these steps:

   a. Click the **Browse for File** button to the right of the **Input File** field .

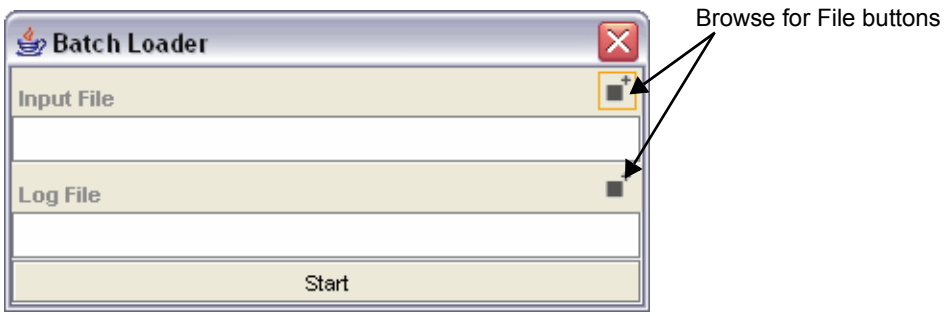


Figure 10.13 Batch Loader window

The browse window opens (*Figure 10.14* on page 157).

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

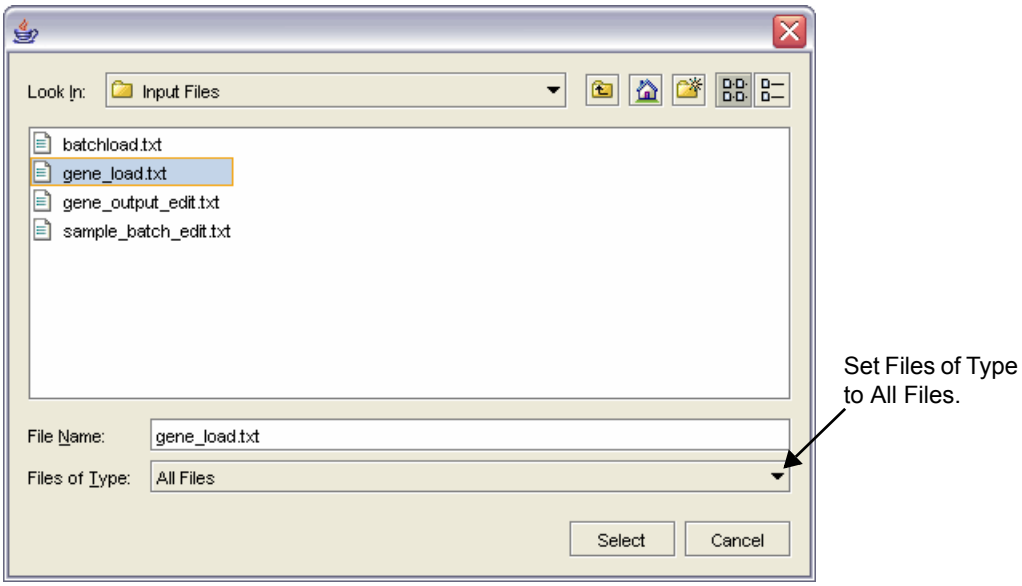


Figure 10.14 Browse window for batch load files

c. Browse for an input file.

d. Click **Select** to close the browse window.

The input file path and name appear in the **Input File** field.

e. Repeat steps *5. a.* through *5. d.* to browse for an output log file.

Note: If you have not already created an output file, you can browse to a specific directory and then type a new name in the **File Name** field. Be sure to append the .txt extension to the name.

6. In the **Batch Loader** window, ensure that both the **Input File** and **Log File** fields show a file path and name (*Figure 10.15*).

7. Click the **Start** button in the bottom of the window.

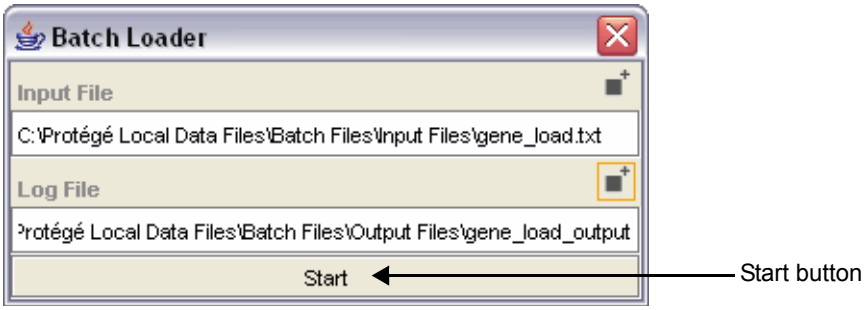


Figure 10.15 Batch Loader window with completed file paths and names

A progress bar confirms that the batch load has begun. When the process is finished, a message window confirms the number of completed actions.

8. Close the message window.

The names of all newly loaded classes appear in the **Batch Loader** display area (*Figure 10.16*).

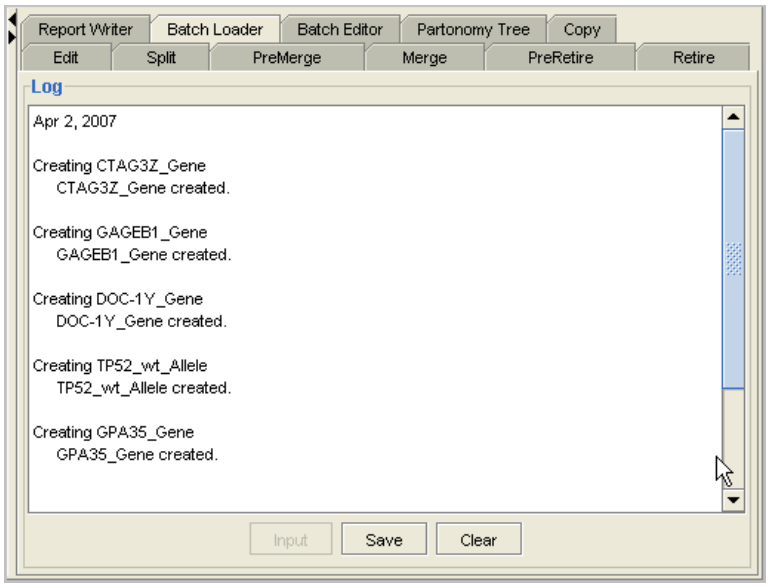


Figure 10.16 Newly imported batch load

9. To find the new classes in the Class Hierarchy, expand the branch for the parent class used in the batch load file.

Note: Although the **Save** button at the bottom of the **Batch Loader** subtab is still active, you do not need to click the button to save the new classes. They have already been saved to the database. You can, however, save the batch output to an external file as a record of the classes that you loaded.

Loading a Batch of Edited Classes

Batch editing is useful when you have a large number of classes and prefer to edit them outside of Protégé. After editing the classes in an external application such as Notepad or Excel, you can import the edited classes into Protégé.

About the Input File Format for a Batch Edit

The application that you use to edit a batch of classes should be capable of saving the edited file in tab-delimited text format (*.txt*). Once you have a text version, you are ready to import the file into Protégé.

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 10.2* describes the fields.

Field No.	*Accepted Values*	*Description*
1	Concept identifier (case sensitive)	For the current configuration, use a class name. In future configurations, this value may be a code.
2	new | edit | delete (case-sensitive)	Use one of these three accepted values to describe the action to be performed on the data. **Note:** If the value in Field 3 is *parent*, the value in this field must be **new** or **delete**.
3	property | role | parent | association	Use one of these four accepted values to describe the property or parent class being modified. **Note:** If the value in this field is *parent*, the value in Field 2 must be **new** or **delete**.
4	Property name or parent name	The value used here is determined by the value in Field 3.
5	See description.	This field is required for a new, edit, or delete action, except when the value in Field 3 is *parent*. For an edit action, this field stores the existing (unedited) value.
6	See description.	For an edit action, this field stores the new (edited) value. Otherwise, it is not required.

Table 10.2 Fields and values for a batch edit file

Note: Qualified properties such as FULL_SYN require that the XML tags of the sub-elements be included as a value. For example, a new FULL_SYN property for Olfactory_Cistern might show the following value in Field 5:

<term-name>name</term-name><term-group>PT</term-group>
<term-source>NCI</term-source>

About the Batch Editor Subtab

The **Batch Editor** subtab manages the process of importing batch-edit files. *Figure 10.17* shows the layout of this subtab.

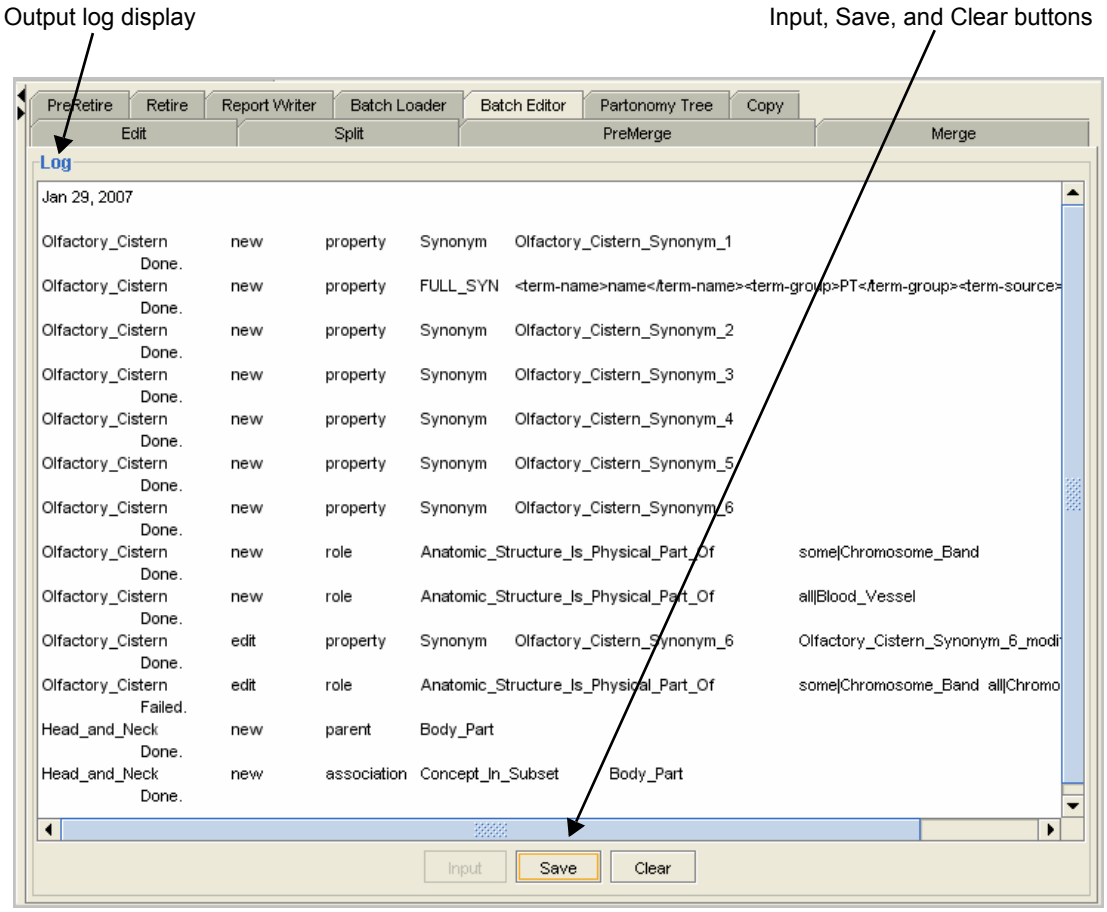


Figure 10.17 Batch Editor subtab

Table 10.3 describes the features of the **Batch Edit** subtab.

Feature	***Description***
Output log display	Shows the log status (the status of the batch edit load process).
Input button	Opens a window in which you can specify an input file and an output log file.
Save button	Saves the log to an ASCII file and confirms the file creation.
Clear button	Clears the display area.

Table 10.3 Batch Editor subtab features

Using the Batch Editor Subtab to Import a Batch of Edited Classes

To import a batch of edited classes, follow these steps:

1. Using a text editor such as Notepad, prepare and save a tab-delimited input file such as the example shown in *Figure 10.18*.

   The file should be structured as described in *Table 10.2* on page 159.

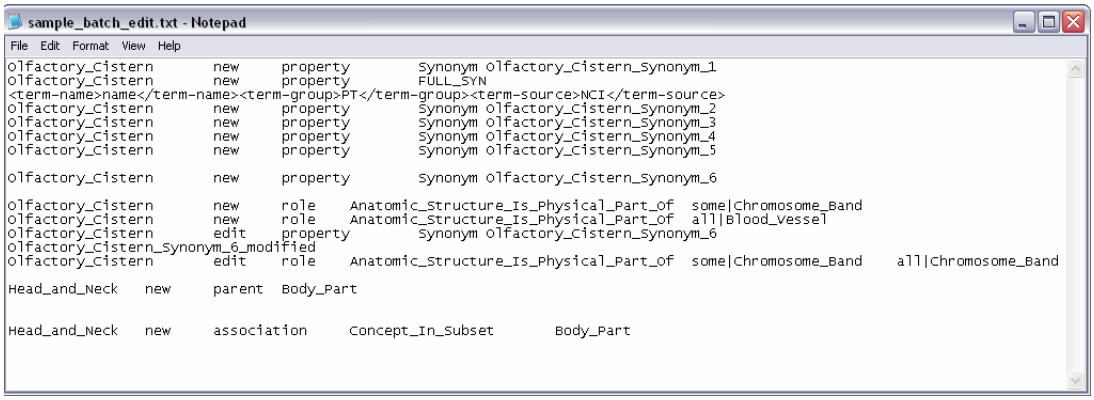


Figure 10.18 Tab-delimited input file for batch load

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

   Note: In this step, the class you select has no bearing on the batch edit procedure. You are performing this step because you cannot use the Batch Editor subtab unless something other than *owl:Thing* is selected in the Class Browser.

3. Click the **Batch Editor** subtab.

   As shown in *Figure 10.19*, the subtab is initially blank until you set up and run the batch edit load process.

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

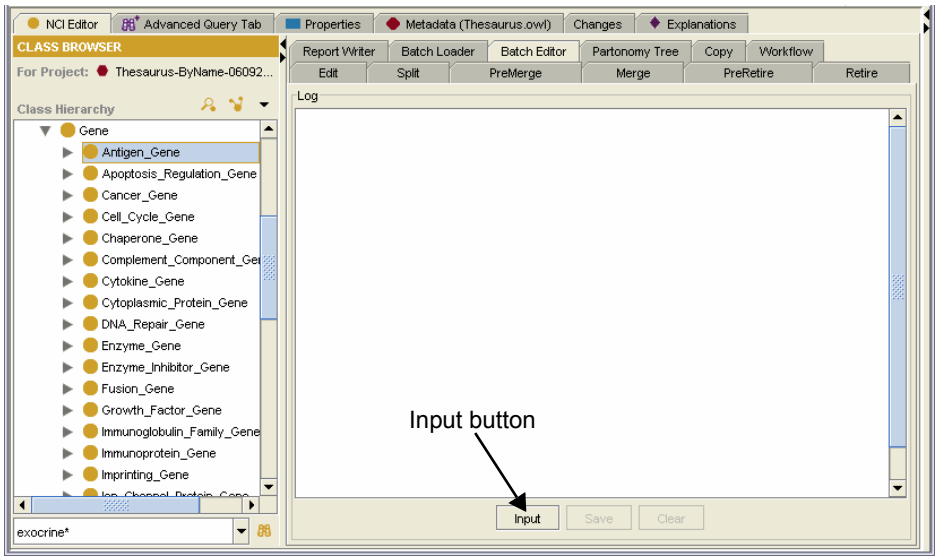


Figure 10.19 Batch Editor subtab

The **Batch Editor** window opens (*Figure 10.20*).

5. Browse for input and output log files by following these steps:

   a. Click the **Browse for File** button to the right of the **Input File** field .

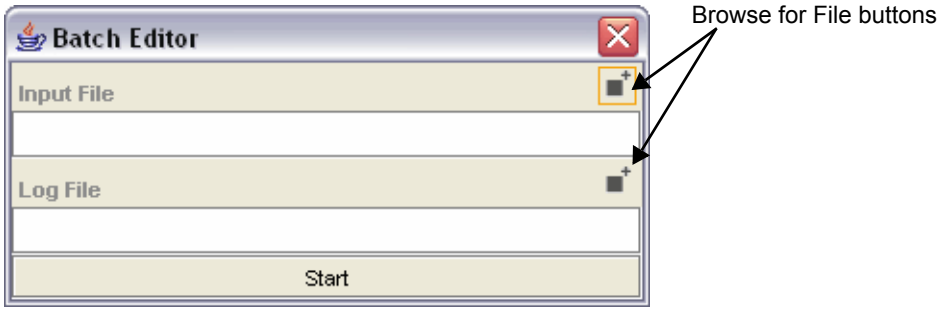


Figure 10.20 Batch Editor window

   The browse window opens (*Figure 10.21*).

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

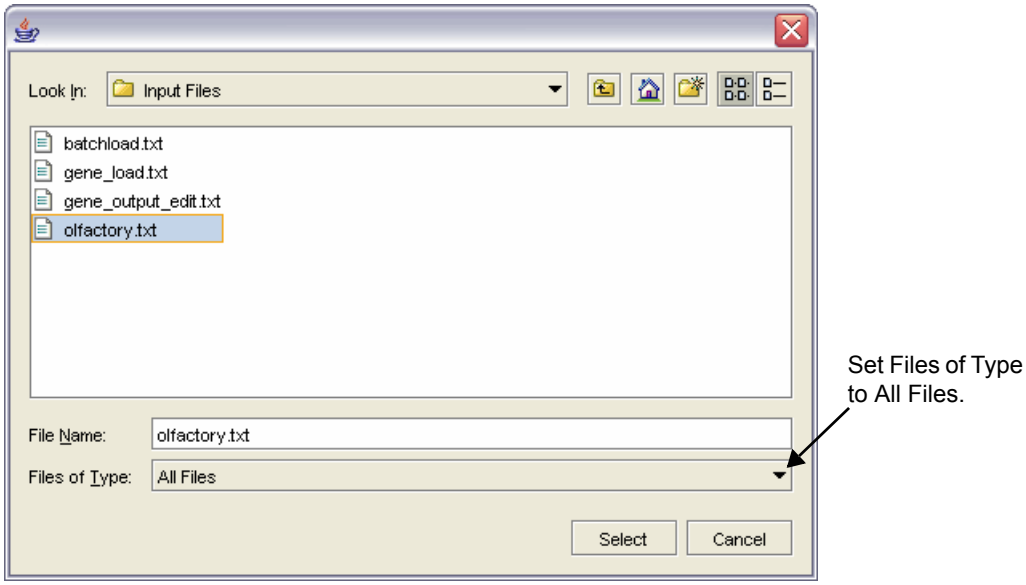


Figure 10.21 Browse window for batch edit files

   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 a log file.

Note: If you haven't already created an output file, you can browse to a specific directory and then type a new name in the **File Name** field. Be sure to append the .txt extension to the name.

6. In the **Batch Editor** window, ensure that both the **Input File** and **Log File** fields show a file path and name (*Figure 10.22*).

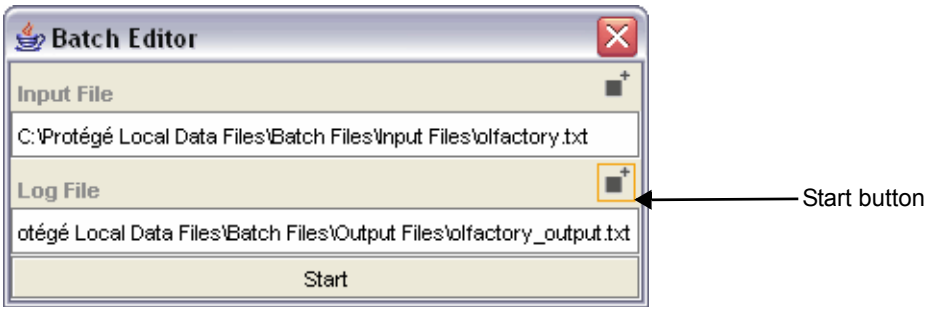


Figure 10.22 Batch Editor window with completed file paths and names

7. Click the **Start** button in the bottom of the **Batch Editor** window.

   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.

8. Close the message window.

   The names of all newly loaded classes appear in the **Batch Loader** display area (*Figure 10.23*).

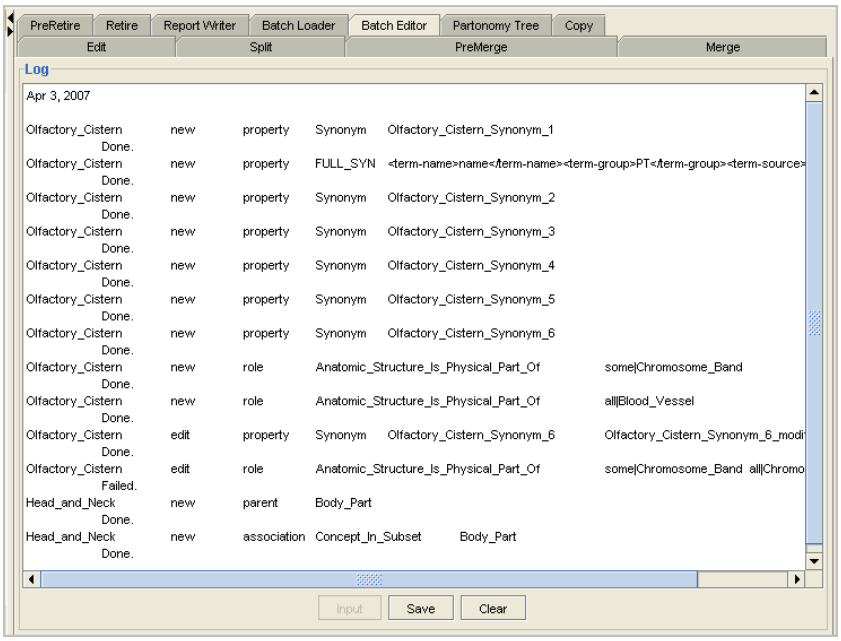


Figure 10.23 Newly imported batch edit

9. To find the new classes in the Class Hierarchy, expand the branch for the parent class used in the batch load file.

Note: Although the **Save** button at the bottom of the **Batch Loader** subtab is still active, you do not need to click the button to save the newly edited classes. The changes have already been saved to the knowledge base. You can, however, save the batch output to an external file as a record of the batch edits that you loaded.

From Here...

Workflow Procedures for NCI Editors on page 165

Chapter 11

Workflow Procedures for NCI Editors

This chapter provides an overview of BiomedGT and the workflow processes and procedures used by external collaborators and NCI terminology curators. Although the procedures in the last section are for NCI editors (non-managers) only, the overview of the workflow process includes those stages for which a workflow manager has primary responsibility.

Topics in this Chapter

- *Overview of BiomedGT* on this page
- *Overview of Internal NCI Workflow* on page 168
- *Workflow Procedures* on page 172

Overview of BiomedGT

About BiomedGT

Purpose

Biomedical terminologies are typically large, standalone terminologies that cover many domains in one structure and use a centralized editing model. Recognizing that this type of terminology no longer adequately supports the needs of the biomedical research community, the NCI Center for Bioinformatics (NCICB) and the cancer Biomedical Informatics Grid (caBIG™) have launched the Biomedical Grid Terminology (BiomedGT).

BiomedGT is an open, collaboratively developed terminology for translational research. This new terminology builds on the strengths of the NCI Thesaurus, including concept orientation, description logic, and public accessibility. While the initial content of BiomedGT is based on the NCI Thesaurus, the terminology has been restructured to

facilitate open content development. The goal is to evolve BiomedGT into a set of smaller federated sub-terminologies called *domain namespaces*, which cover specific areas of biomedical knowledge. As experts in the relevant research communities stabilize the content of the various domain namespaces, new content will be incorporated into the description logic-based, federated BiomedGT by the NCICB EVS and volunteer editors.

BiomedGT is intended to be the first of a new generation of biomedical terminologies that are created using an open content development approach. Such an approach empowers the wider biomedical research community to directly participate in extending and refining the terminology upon which they depend. The approach also enables maximum community participation without requiring that contributors learn complex terminology curation software and without sacrificing the quality assurance standards needed to manage biomedical terminologies.

Audience

BiomedGT is intended to address the needs of biomedical researchers who work in both basic and clinical research. Researchers can acquire content by

- downloading it in multiple formats such as OWL and RRF;
- using Web-enabled applications; or
- using Application Programming Interfaces (APIs).

About the BiomedGT Wiki

The BiomedGT wiki (http://biomedgt.org) is a Web-based, collaborative terminology authoring tool being developed by the NCICB and Apelon, Inc. *Figure 11.1* shows the home page of the wiki.

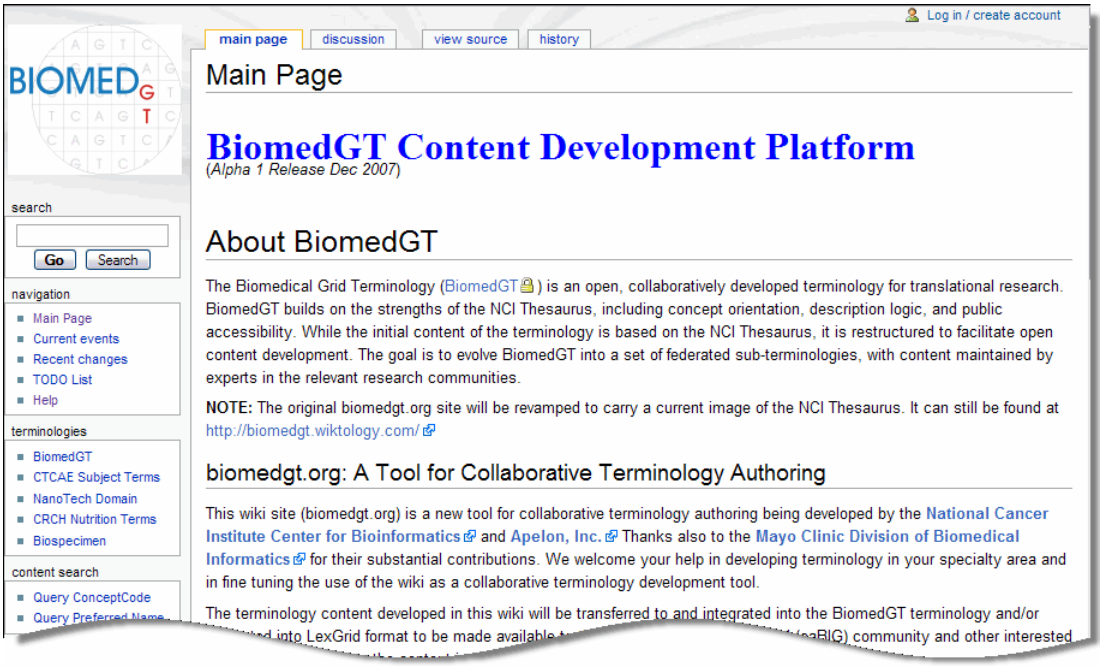


Figure 11.1 BiomedGT Home page

The BiomedGT wiki pulls terminology data from a read-only database and displays it on a Web page. This facilitates the process described in *Table 11.1*.

Stage	***Action***	***Responsibility***
1.	Review the read-only, published BiomedGT terminology.	Outside collaborator
2.	Using the provided Web form, submit a *structured* or *unstructured* proposal for any of the following reasons: • To suggest changes to existing concepts (structured) • To suggest the addition of new concepts (structured) • To direct NCI curators to a specific wiki discussion page by providing the URL (unstructured). Structured proposals are also called *packages* because they often include batches of concepts. Unstructured proposals typically refer to a wiki discussion page that might discuss specifics about a concept and then make a recommendation.	Outside collaborator
3.	Import packages and incorporate them into the vocabulary as needed. This stage has its own internal workflow processes and procedures as described in *Overview of Internal NCI Workflow* on page 168.	NCI BiomedGT curators
4.	Communicate with submitting collaborators to clarify suggestions or explain the outcome of decisions regarding proposals.	NCI BiomedGT curators
5.	Publish updated read-only database and display it on the Web page.	BiomedGT administrator

Table 11.1 Workflow process: BiomedGT to Protégé

For more information about BiomedGT, see the following online resources:

- BiomedGT project on GForge:
  http://gforge.nci.nih.gov/projects/biomedgt/
- EVS section of the NCICB web site:
  http://ncicb.nci.nih.gov/NCICB/infrastructure/cacore_overview/vocabulary
- BiomedGT wiki site:
  http://biomedgt.org

Overview of Internal NCI Workflow

As biomedical research community experts increasingly use the BiomedGT wiki for collaborative terminology development, NCI curators need a managed, organized workflow process. This section discusses the process established for the NCI.

Primary Goals

Following are the primary goals of the BiomedGT/Protégé workflow:

- To provide capability that supports workflow manager and modeler roles, including assignments, tracking, auditing, and reporting
- To provide a means for extracting new terms from the BiomedGT wiki and incorporating them into assignments for NCI Protégé modelers
- To provide status and auditing reports to help management assess modeling progress.

Workflow Roles

The internal workflow for NCI Protégé consists of two primary roles:

- **Manager**

  A manager oversees the coordination and distribution of work assignments. The assignments may originate from the BiomedGT wiki or from other members of the editing team.

  The manager performs the following tasks:

  - Imports packages from the BiomedGT wiki
  - Creates assignments for editors
  - Tracks assignments
  - Incorporates completed assignments into the NCI Thesaurus.

  Note: The manager may also perform routine editing tasks.

- **Modeler (Editor)**

  A modeler is an NCI editor who performs routine editing tasks. Modelers cannot import packages, create assignments, or delete assignments. However, they can make *suggestions*, which are similar to assignments but which do not become assignments until a manager has reviewed them.

  The modeler performs the following tasks:

  - Receives assignments and accepts or rejects them
  - Incorporates new or changed terminology into the NCI Thesaurus using the NCI Protégé work environment
  - Maintains responsibility for the assigned concept, including any follow-up communication to the collaborator who submitted the concept for consideration
  - Marks assignments as complete and submits them to the manager.

Workflow Stages

Table 11.2 provides an overview of the internal workflow stages and the responsible role for each stage.

Note: Each submitted item is called a *workflow item* until it has been assigned to a modeler. It then becomes an *assignment*.

Stage	*Action*	*Responsibility*
1.	Log into Protégé.	Manager
2.	Using the NCI Workflow tab, import packages from the BiomedGT wiki into Protégé.	Manager
3.	Create workflow items with a status of CREATED.	System
4.	Review each new workflow item and assigns it to a modeler.	Manager
5.	Change the status of the workflow item to ASSIGNED.	System
6.	Log into Protégé.	Modeler
7.	Check the **NCI Workflow** tab for assignments.	Modeler
8.	Accept an assignment.	Modeler
9.	Change the status of the assignment to ACCEPTED.	System
10.	Examine the proposed and existing (if any) concepts.	Modeler
11.	Perform one of the following tasks: • Edit and save the existing concept (if any); if completed, mark the assignment as COMPLETED. • Add notes to the assignment regarding the proposed concept and existing concept (if any); if completed, mark the assignment as COMPLETED. • Reject the assignment.	Modeler
12.	Change the status of the assignment to UPDATED (if not marked COMPLETED), COMPLETED, or REJECTED, depending on the action in step 11.	System
13.	Review completed and rejected assignments; address rejected assignments according to the nature of the proposal.	Manager

Table 11.2 BiomedGT wiki process for curating terminology

Note: In addition to work items generated by BiomedGT wiki imports, the manager also handles work items that have been submitted internally by modelers. The system assigns a status of SUGGESTED to internally submitted work items.

About the NCI Workflow Tab

The **NCI Workflow** tab is the major workflow component used by both managers and modelers. The tab layout is a table grid format, with each row representing a workflow item or assignment (*Figure 11.2*).

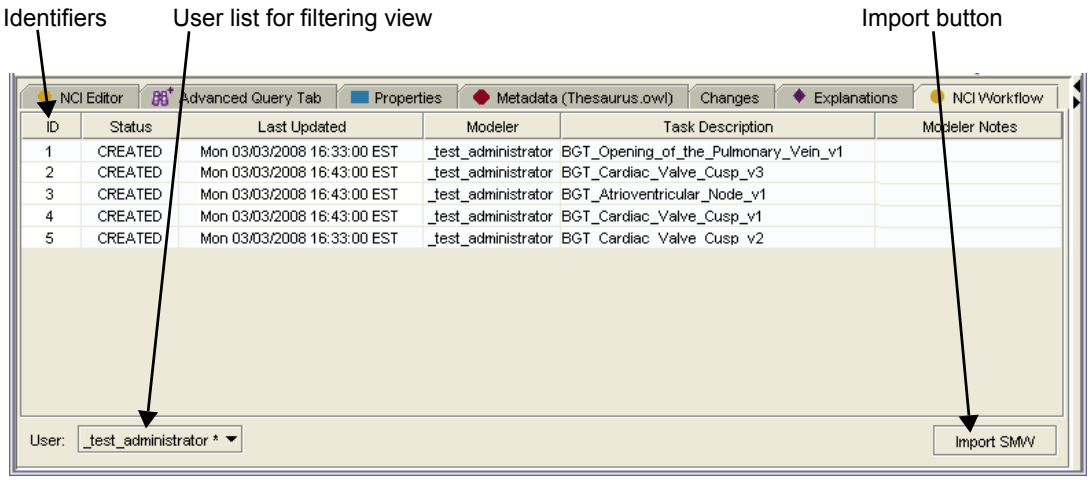


Figure 11.2 NCI Workflow tab

Protégé determines what the **NCI Workflow** tab displays based on the privileges of the user who is currently logged in:

- Managers can see all assigned and unassigned workflow items and can identify the assignee and status of each assigned item. They can also filter the assignments by user.
- Modelers see only workflow items that have been assigned to them.

Table 11.3 describes the fields that identify each workflow item and assignment.

Field	***Description***
ID	Shows a sequential number that is assigned to each workflow item. Once a number is assigned to an item, it remains the same, even if the same item is re-imported.
Status	Shows the current status of a workflow item or assignment: • CREATED: The manager has imported a workflow item but has not yet assigned it. • SUGGESTED: A modeler has created a workflow item that is relevant to an existing concept. • ASSIGNED: The manager has assigned a workflow item to a modeler but the modeler has not yet accepted it. • ACCEPTED: The modeler has accepted the assignment. • UPDATED: The modeler saved a change to a class or added a note but did not mark the assignment as COMPLETED. • COMPLETED: The modeler has completed the assignment.

Table 11.3 NCI Workflow tab fields

Field	Description
Last Updated	Shows the weekday, date, and time in the following format: Fri 03/07/2008 16:34:00 EST
Modeler	Initially shows the name of the user (manager) who imported the workflow item. Once assigned, this field displays the name of the assignee.
Task Description	For structured proposals, shows the name of a concept or a URL pointing to a wiki discussion page. Concept names are preceded by the letters BGT to note that the concept has been imported from the BiomedGT wiki.
Modeler Notes	Shows the most recent note added to a workflow item or assignment, followed by an ellipsis (...). If you hover the mouse pointer over the date and time, pop-up text shows the note text starting with the most recent entry.

Table 11.3 NCI Workflow tab fields (Continued)

The grid area of the **NCI Workflow** tab supports right-click commands. *Table 11.4* describes the available commands. Some commands may appear as unavailable depending on your user privileges.

Command	Description
Edit Concept	Displays the proposed concept and existing concept (if any) on the **Workflow** subtab. You can then edit, add notes to, or reject the concept.
Accept Assignment	Enables you to accept an assignment.
Modify Assignment	Opens a window in which you can add notes and view previous notes.
Complete Assignment	Enables you to change the status of an assignment to COMPLETED.
Delete Assignment	Enables you to delete an assignment, depending on your user privileges and the status of the assignment: • Managers can only delete assignments with a status of CREATED, SUGGESTED, or COMPLETED. • Modelers can only delete SUGGESTED assignments.
Cancel	Closes the shortcut menu.

Table 11.4 NCI Workflow tab - right-click commands

Workflow Procedures

This section provides the steps for the main procedures that NCI editors (modelers) typically perform. Most of the procedures originate from the NCI Workflow tab.

Note: For more information about modeler procedures in relation to the overall NCI internal workflow process, see *Overview of Internal NCI Workflow* on page 168.

Topics in this Section

- *Accepting a Workflow Assignment* on this page
- *Editing an Assigned Concept* on page 13
- *Adding Notes to an Assignment* on page 14
- *Rejecting an Assignment* on page 15
- *Completing an Assignment* on page 16
- *Suggesting an Assignment* on page 16

Accepting a Workflow Assignment

Before you can work on an assignment, you must first accept it. To accept an assignment, follow these steps:

1. Log in to Protégé.
2. Click the **NCI Workflow** tab.
3. In the list of workflow items, look for new items with a status of ASSIGNED (*Figure 11.3*).

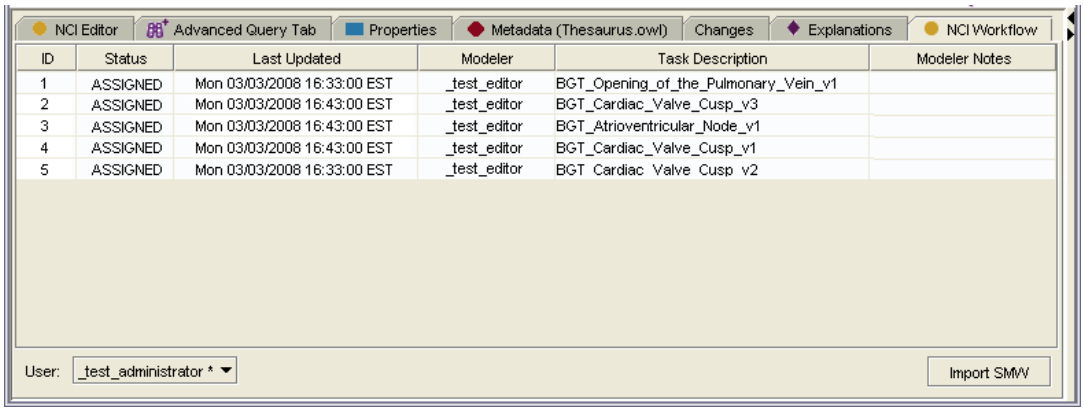


Figure 11.3 Workflow items with ASSIGNED status

4. Right-click anywhere on the row that displays the assignment you want to accept.
5. In the shortcut menu, select **Accept Assignment**.

   The **Status** column now shows a status of ACCEPTED.

Tip: You can always use the NCI Workflow tab to periodically check for new workflow assignments and verify the status of your existing assignments.

Editing an Assigned Concept

Prerequisite task: *Accepting a Workflow Assignment* on page 168

After accepting an assignment, you need easy access to the related concepts. Regardless of whether you are working on an assignment for the first time or returning to an assignment that you have not yet completed, you can access the related concepts from the **NCI Workflow** tab.

To begin work or resume work on an assignment, follow these steps:

1. Click the **NCI Workflow** tab, then locate the assignment requiring your attention.
2. Right-click anywhere on the row that displays the assignment.
3. In the shortcut menu, click **Edit Concept**.

   The result depends on the origin of the assignment, as described in *Table 11.5*.

Assignment Origin	***Result***
Imported wiki package with proposed new concept or proposed changes to an existing concept...	Protégé displays the **Workflow** subtab with the *proposed* concept in the upper pane and the *existing* concept (if any) in the lower pane (*Figure 11.4*).
Imported wiki package with reference to a wiki discussion page...	A pop-up message window displays the URL of the discussion page. Eventually, the page content will be displayed within the Protégé work environment.
Suggested by an NCI editor...	Protégé displays the **Edit** subtab > **Basic Data** subtab with details for the existing concept.

Table 11.5 Results of selecting different types of assignments

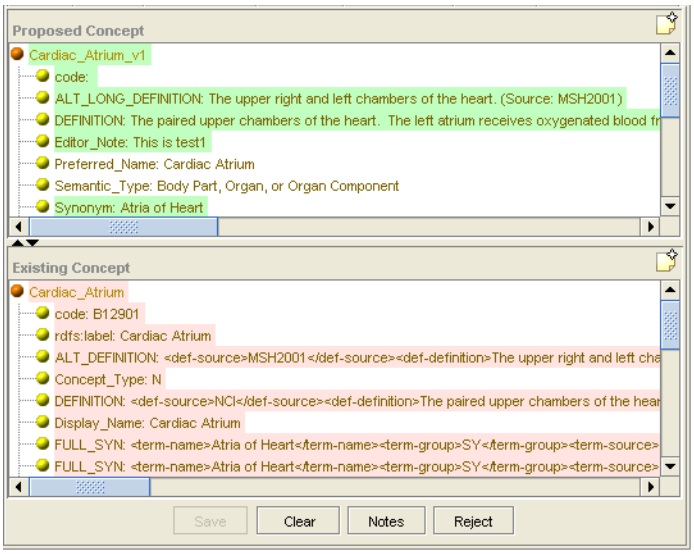


Figure 11.4 Assignment from wiki import - proposed and existing concepts

Adding Notes to an Assignment

Prerequisite task: *Accepting a Workflow Assignment* on page 168

Using the NCI Workflow Tab

Protégé provides a window in which you can add notes to an assignment or read previous notes that you or others have already added.

To add notes to an assignment, follow these steps:

1. Click the **NCI Workflow** tab, then locate the assignment requiring notes.
2. Right-click anywhere on the row that displays the assignment.
3. In the shortcut menu, click **Modify Assignment**.

   A window opens with the assignment number in the title bar. The top pane shows any previously added notes, with the most recent note at the top. The bottom pane enables you to add a new note (*Figure 11.5*).

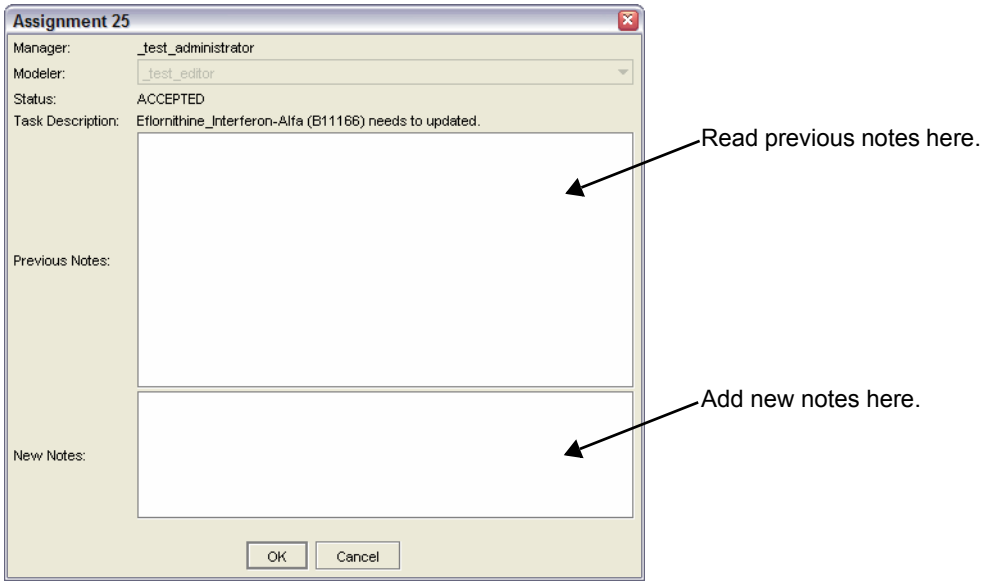


Figure 11.5 Window for adding notes to an assignment

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

Using the Workflow Subtab

Prerequisite task: *Editing an Assigned Concept* on page 169

To add notes while viewing proposed and existing concepts on the **Workflow** subtab, follow these steps:

1. Click the **Notes** button at the bottom of the subtab.

   The **Modelers Notes** window opens. The top pane shows any previously added notes, and the bottom pane enables you to add a new note.

2. Once you finish adding new notes, click **OK**.

Rejecting an Assignment

You may sometimes need to reject an assignment in cases such as the following:

- The proposed new concept already exists but was not found by the wiki user.
- The concept is deemed to be out of scope.

Note: Before rejecting an assignment, you or your workflow manager need to discuss the proposed new concept or concept changes with the outside collaborators who submitted the proposal. For example, if the requested modeling does not classify correctly, you may need to clarify the intended usage. You may find after asking for clarification that the concept can be modified to conform to NCI editing guidelines.

Prerequisite tasks:

- *Accepting a Workflow Assignment* on page 168
- *Editing an Assigned Concept* on page 169

If you cannot complete an assignment, or if the assignment should be assigned to someone else, you can reject that assignment.

To reject an assignment, follow these steps:

1. Ensure that the assignment is open for editing on the **Workflow** subtab.

   The subtab displays the proposed concept and the existing concept (if any).

2. Click the **Reject** button at the bottom of the subtab.

   The **Rejection Notes** window appears.

3. (Required) Add notes to the **New Notes** section (lower section) of the window.
4. To verify that the assignment status has been changed, click the **NCI Workflow** tab.

   The assignment now shows a REJECTED status.

Note: In the current release, you can reject only assignments that originate from a wiki package and include a proposed new concept or proposed changes to an existing concept. You cannot reject assignments that (1) originate from a wiki page and refer to a wiki discussion page or (2) originate as internal suggestions.

Completing an Assignment

Prerequisite tasks:

- *Accepting a Workflow Assignment* on page 168
- *Editing an Assigned Concept* on page 169

Once you have completed an assignment, you need to mark it as completed.

To mark an assignment as completed, follow these steps:

1. Click the **NCI Workflow** tab, then locate the assignment that you have completed.
2. Right-click anywhere on the row that displays the assignment.
3. In the shortcut menu, click **Complete Assignment**.

   The **Status** column now shows a status of COMPLETED.

Suggesting an Assignment

You can suggest an assignment that is relevant to an existing concept. NCI Protégé creates a new workflow item with a status of SUGGESTED. The workflow manager can then assign the item to a modeler.

To suggest an assignment, follow these steps:

1. Select the **Copy** subtab.
2. Using the Class Browser, locate the class for which you want to suggest an assignment.
3. Drag the class into the upper pane of the **Copy** subtab.
4. Right-click any property in the upper pane, then select **Create Workflow Task** from the shortcut menu.

   A message window confirms that a new assignment has been created. The message includes the assignment number (*Figure 11.6*).

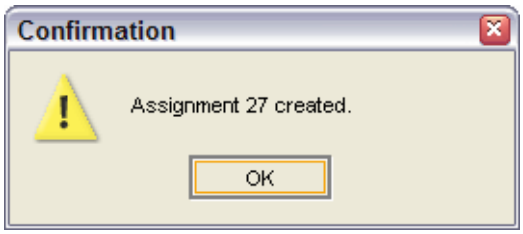


Figure 11.6 Message confirming creation of suggested assignment

5. To verify that the assignment has been created, click the **NCI Workflow** tab.
6. Locate the assignment by number.

   In the row for the new assignment, the **Status** column shows a status of SUGGESTED.

Appendix A

TDE vs. Protégé Terminology

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

TDE Term	***Protégé Term or Description***
All	allValuesFrom (all, only, universal quantifier)
Association	AnnotationProperty of type ObjectProperty
Bottom	Nothing
Concept	• Class (set semantics) • Individuals
Defined	Necessary & Sufficient conditions asserted
Kind	Top-level disjoint class
Pick Lists	Enumerated Datatype AnnotationProperty
Primitive	Necessary conditions asserted
Property	AnnotationProperty (AnnotationProperty of type DatatypeProperty)
Qualifier	XML - as in TDE when we first began using it
Role	• ObjectProperty • DatatypeProperty
Role Expression	Restriction on property
Some	someValuesFrom (some, existential quantifier)
Top	Thing

Table A.1 TDE terms and their Protégé equivalents

INDEX

A

adding
- association 117 – 120
- FULL_SYN 102
- notes to assignment 174
- parent class 92
- restriction 106 – 109
- role group 113 – 116

advanced query
- building 57 – 60
- data types from SMI 79
- examples
  - association-based 70
  - combined property 64 – 66
  - complex property 62 – 63
  - restriction-based 67 – 70
  - simple property 61
- syntax 59

Advanced Query tab
- about 40, 57
- configuring 75 – 78

advanced search
- *See* advanced query

advanced vs. simple search 53

annotation property
- *See* property

assignment
- accepting 172
- adding notes to 174
- completing 176
- editing 173
- rejecting 175

association
- adding 117 – 120
- searching by 70

axioms in OWL 21

B

Basic Data subtab 85

batch
- edit file format 159
- load file format 154
- loading classes for editing 154 – 158
- loading edited classes 159 – 163

Batch Editor subtab 45, 160

Batch Loader subtab 45, 154

BiomedGT overview 165 – 167

bug report, submitting 49

C

Changes tab 41

class
- adding parent 93
- cloning 133 – 135
- creating 89 – 92
- deleting parent 96
- descriptions in OWL 18
- examining information about 84 – 87
- finding pre-retired in hierarchy 143
- generating report for 147 – 153
- merging with second class 127 – 133
- modifying parent 95
- overview 81 – 83
- pre-merging with second class 129 – 131
- pre-retiring 138 – 143
- retired, finding in hierarchy 131
- retiring 144 – 145
- splitting 123 – 127
- treeing 92 – 94
- viewing by part_of relations 97

Class Browser 42 – 44

classes
- editing two simultaneously 120
- loading batch for editing 154 – 158
- loading batch of edited 159 – 163

cloning class 133 – 135

Collaboration panel
- about 41

command line window
- *See* Console

concepts in description logic 17

conditions, necessary and sufficient 21

Console
about 46
capturing troubleshooting information 48
overview 46
setting display capacity 47
contacting support 5
copying class 133 – 135
Copy subtab 45
creating class 89 – 92

D

definition
modifying 104
overview 82 – 83
deleting
FULL_SYN 104
parent class 96
restriction 112
description logics
and OWL 15 – 26
in NCI Thesaurus 23
disjointness axiom 21
DOS window
See Console
downloading
installation file 28
Thesaurus subset file 33

E

editing two classes simultaneously 120
Edit subtab 44, 84
Enterprise Vocabulary Services
See EVS
equivalency axiom 21
EVS
overview 7 – 9
related online resources 11 – 12
existential qualifier 22
Explanations tab 41
exporting search results 72 – 74

F

file format
for batch edit 159
for batch load 154
files
available on GForge for Protégé 27
downloading installation 28
finding pre-retired class in hierarchy 143
finding retired class in hierarchy 131
first-order predicate logic 14
frame-based representations 13
FULL_SYN
about 82
adding 102
deleting 104
modifying 103
full synonym
See FULL_SYN

G

generating
partonomy tree 98
report for selected class 147 – 153
GForge
available Protégé files 27
submitting bug report to 49

I

installing Protégé 27 – 29
interface overview 40 – 45

K

knowledge representation
first-order predicate logic 14
frame-based 13

L

logging in to Protégé 30 – 32

M

Merge subtab 44, 131
merging classes 132
Metadata tab 40
Metathesaurus (NCI)
See NCI Metathesaurus
Metathesaurus (UMLS), overview 10 – 11
modifying
definition 104
FULL_SYN 103
parent class 95
restriction 109 – 112

N

NCI Editor tab
about 40, 42 – 45
subtabs of 44 – 45
NCI Thesaurus
about 8
and OWL 24
description logics in 23
semantic model 13 – 26
NCI Workflow tab 170

about 41
necessary and sufficient conditions 21
notes, adding to assignment 174

O

OWL
and description logics 15 – 26
and NCI Thesaurus 24
axioms used in 21
class descriptions 18 – 20
levels of expressivity 17
property restrictions 22
qualifiers in 22

P

parent class
adding 93
deleting 96
modifying 95
partonomy tree, generating 98
Partonomy Tree subtab 45, 97
PreMerge subtab 44, 128
pre-retired class, finding in hierarchy 143
PreRetire subtab 44, 138
previous version of Protégé, uninstalling 29
Properties subtab 87
Properties tab (main) 40
property
asserting 101 – 104
definition 82
FULL_SYN 82
restrictions in OWL 22
Protégé application
installing 27 – 29
interface overview 40 – 45
logging in 30 – 32
overview 39
uninstalling 29
using locally 32 – 37

Q

qualifiers, existential and universal 22
query
See advanced query

R

relation, asserting 106 – 116
Relations subtab 86
report, generating for selected class 147 – 153
Report Writer subtab 44, 147
restriction
adding 106 – 109
deleting 112
modifying 109 – 112
retired class, finding in hierarchy 131
retirement process for classes 137
Retire subtab 44, 144
retiring class 144 – 145
Review window 88
role, in description logic 17
role group, adding 113 – 116

S

search
advanced
See advanced query
by association 70
results
exporting to external file 72 – 74
viewing and selecting 56
simple, performing 54 – 55
simple vs. advanced 53
Semantic MediaWiki
See BiomedGT overview
SMI query data types 79
Split subtab 44
Split tab 124 – 126
splitting class 123 – 127
subsumption axiom 21
support, contacting 5

T

Thesaurus
See NCI Thesaurus
treeing class 92

U

UMLS Metathesaurus, overview 10 – 11
uninstalling Protégé application 29
universal qualifier 22

V

version, uninstalling previous 29

W

workflow overview
BiomedGT to Protégé 167
internal NCI process 168 – 171
workflow procedures (NCI editors) 172 – 176
accepting assignment 172
adding notes to assignment 174
completing assignment 176
editing assignment 173
rejecting assignment 175

Workflow subtab 45