

# The SeqHound Manual

Release 3.2

(March 31<sup>st</sup>, 2005)

## Authors

Ian Donaldson, Katerina Michalickova, Hao Lieu, Renan Cavero, Michel Dumontier, Doron Betel, Ruth Isserlin, Marc Dumontier, Michael Matan, Rong Yao, Zhe Wang, Victor Gu, Elizabeth Burgess, Kai Zheng, Rachel Farrall

## Edited by

Rachel Farrall and Ian Donaldson

## Table of Contents

1. What is SeqHound?.....	8
<i>How to contact us.</i> .....	9
<i>Who is SeqHound?</i> .....	10
<i>About this manual.</i> .....	11
Conventions.....	12
2. How to get started .....	13
3. Using the SeqHound remote API.....	14
<i>How to get started with the remote API.</i> .....	14
<i>An overview of the SeqHound API.</i> .....	16
SYSTEM INITIALIZATION, FINISHING AND TESTING.....	17
GENBANK ID CONVERSIONS .....	17
SEQUENCE AND STRUCTURE CHECKS .....	18
SEQUENCE FETCH ASN.1 BIOSEQ.....	19
SEQUENCE FETCH ASN.1 BIOSEQSET .....	19
SEQUENCE FETCH ASN.1 SEQENTRY .....	19
SEQUENCE FETCH FASTA.....	20
DEFLINE FETCH.....	20
SEQUENCE FETCH GENBANK FLAT FILE .....	20
STRUCTURE FETCH.....	20
SEQHOUND LINKS .....	21
COMPLETE GENOME ITERATORS.....	22
REDUNDANT (EQUIVALENT) SEQUENCES.....	22
TAXONOMY.....	23
SEQUENCE NEIGHBOURS .....	23
COMPLETE GENOME ANALYSIS .....	24
FUNCTIONAL ANNOTATION.....	24
GO HIERARCHY .....	25
RPS BLAST DOMAINS .....	26
DEPRECATED FUNCTIONS .....	26
<i>Underlying HTTP calls.</i> .....	28
<i>The complete SeqHound API description.</i> .....	29
<i>Language support for API functions.</i> .....	30
<i>Using the remote API for C</i> .....	31
<i>Using the remote API for C++*</i> .....	32
<i>Using the remote API for Java</i> .....	33
Special Notes.....	33
System Requirements.....	36
Using the pre-compiled binary distribution. ....	37

Test and verify your binary SeqHound Java API installation.....	39
Compile, Build and Install Seqhound Java .....	40
Using the SeqHound Java API .....	43
API Documentation.....	44
Configuration .....	45
Bugs.....	48
<i>Using the remote API for PERL.....</i>	<i>49</i>
Troubleshooting .....	51
Documentation .....	51
Bugs.....	52
<i>Using the remote API for Bioperl.....</i>	<i>53</i>
Prerequisites .....	53
Installation.....	53
Documentation .....	54
Bugs.....	55
4. Setting up SeqHound locally. ....	56
4.1 Overview.....	56
4.2 SeqHound system requirements.....	57
OS and hardware architecture .....	57
Memory (RAM) .....	57
Hard Disk .....	58
Source code and executables .....	58
Database.....	58
Other Software .....	58
Compiling SeqHound Code yourself. ....	58
Database engine.....	58
Library dependencies .....	58
4.3 Obtaining SeqHound executables and/or code. ....	59
4.4 Compiling SeqHound executables on Solaris.....	61
4.5 Building the SeqHound system on Solaris.....	69
Catch up on SeqHound daily updates .....	87
Setting up daily sequence updates.....	89
Setting up SeqHound servers. Overview.....	95
Trouble-shooting notes.....	99
Error logs .....	99
Recompiling SeqHound.....	99
Restarting the Apache server .....	99
Other useful links.....	100
Parser schedule.....	100
MySQL errors .....	100
5. Description of the SeqHound parsers and data tables by module.....	101
<i>What are modules? .....</i>	<i>101</i>
<i>How to use this section. ....</i>	<i>101</i>

<i>Parser descriptions.....</i>	<i>101</i>
<i>Table descriptions.....</i>	<i>102</i>
<i>An overview of the SeqHound data table structure .....</i>	<i>105</i>
<i>Parsers and resource files needed to build and update modules of SeqHound. ....</i>	<i>106</i>
core module .....	109
mother parser .....	109
update parser .....	112
asndb table .....	116
parti table .....	120
nucprot table.....	123
accdb table .....	126
histdb table.....	133
pubseq table .....	138
taxgi table.....	141
sengi table .....	147
sendb table .....	151
Redundant protein sequences (redundb) module .....	154
redund parser.....	154
redund table.....	156
Complete genomes tracking (gendb) module.....	160
genff parsers:.....	160
chrom parser.....	161
comgen parser .....	163
chrom table.....	165
Taxonomy hierarchy (taxdb) module .....	168
importtaxdb parser .....	168
taxdb table.....	170
gcodedb table .....	174
divdb table.....	178
del table.....	180
merge table.....	181
Structural databases (strucdb) module .....	183
cbmmdb parser.....	183
vastblst parser.....	188
pdbrep parser.....	190
mmdb table.....	192
mmgi table .....	196
domdb table.....	198
Protein sequence neighbours (neighdb) module .....	202
Installing nblast:.....	202
Configuration of nblast environment: .....	203
Running NBLAST .....	204
NBLAST Update Procedure .....	206
nbraccess program* .....	208
BLASTDB table.....	209

NBLASTDB table.....	211
Locus link functional annotations (lldb) module .....	214
llparser.....	214
addgoid parser.....	216
ll_omim table .....	218
ll_go table.....	220
ll_llink table .....	223
ll_cdd table.....	225
GENE module .....	228
parse_gene_files.pl parser.....	228
gene_dbxref table.....	230
gene_genomicgi table .....	232
gene_history table .....	235
gene_info table.....	238
gene_object table .....	241
gene_productgi table.....	244
gene_pubmed table .....	246
gene_synonyms table.....	248
Gene Ontology hierarchy (godb) module.....	250
goparser.....	250
go_parent table.....	252
go_name table .....	254
go_reference table.....	257
go_synonym table .....	259
RPS-BLAST domains (rpsdb) module.....	261
domname parser .....	261
Rpsdb parser.....	262
domname table .....	263
rpsdb table.....	267
Molecular Interaction (MI) module.....	274
MI-BIND parser.....	274
MI_source table .....	278
MI_ints table .....	280
MI_objects table.....	281
MI_obj_dbases table .....	283
MI_mol_types table .....	284
MI_dbases table .....	285
MI_record_types table.....	286
MI_complexes table.....	287
MI_complex2ints table .....	288
MI_complex2subunits table.....	289
MI_complex2subunits table.....	290
MI_refs table.....	291
MI_refs_db table.....	293
MI_exp_methods table.....	294
MI_obj_labels table .....	295

Text mining module .....	297
mother parser .....	297
text searcher parser .....	298
yeastnameparser.pl parser .....	302
text_bioentity table.....	304
text_bioname table.....	307
text_secondrefs table.....	311
text_bioentitytype table.....	314
text_fieldtype table.....	315
text_nametype table .....	316
text_rules table.....	317
text_db table.....	318
text_doc table.....	319
text_docscore table.....	321
text_evidencescore table .....	326
text_method table.....	328
text_point table.....	331
text_pointscore table .....	332
text_result table.....	334
text_resultscore table .....	336
text_search table.....	338
text_searchscore table .....	341
text_rng table .....	343
text_rngresult table.....	345
text_doctax table .....	347
text_organism table.....	349
text_englishdict table .....	351
text_bncorpus table .....	354
text_pattern table.....	356
text_stopword table.....	358
Gene Ontology Association (GOA) module .....	360
goa_parser_cluster.pl .....	361
goa_geneparser_cluster.pl.....	361
seqhound_gigo_cluster.pl .....	361
seqhound_gigoPDB_cluster.pl.....	361
seqhound_gigoCGEN_cluster.pl .....	361
goa_seq_dbxref table .....	364
goa_association table .....	368
goa_reference table .....	371
goa_reference indices.....	372
goa_with table.....	374
goa_with indices .....	376
goa_xdb table.....	377
goa_xdb indices .....	379
goa_gigo table.....	380
goa_gigo indices .....	382

Gene Ontology Module Diagram.....	383
dbxref module .....	385
Table summarizing input files, parsers and command line parameters for dbxref module.....	390
dbxref table .....	392
dbxref indices.....	394
dbxrefsourcecdb table.....	395
dbxrefsourcecdb indices .....	396
Contents of dbxrefsourcecdb table .....	398
6. Developing for SeqHound. ....	400
<i>Open source development.</i> .....	400
<i>Code organization.</i> .....	401
<i>Adding/Modifying a remote API function to SeqHound.</i> .....	405
Overall architecture of the SeqHound system.....	406
<i>Adding a new module to SeqHound.</i> .....	412
Database layer .....	413
Parser layer.....	414
Local API layer (Query layer).....	415
CGI layer.....	415
Remote API layer .....	416
7. Appendices.....	419
Example GenBank record .....	420
Example SwissProt record .....	425
Example EMBL record .....	432
Example PDB record.....	438
Example Biostruc .....	443
GO background material .....	453

\* not available at time of writing

# 1. What is SeqHound?

The simple answer.

SeqHound is a biological sequence retrieval system inspired by *Entrez*. The sequence and structure database content is derived from GenBank and is updated on a daily basis.

SeqHound also stores annotation links for sequences; these include links to Gene Ontology descriptions, MedLine abstracts, the NCBI taxon database, associated structures, redundant sequences, sequence neighbours, conserved domains, data base cross-references, Online Mendelian Inheritance in Man identifiers, LocusLink identifiers and complete genomes.

There are three ways to access information in SeqHound.

1. As a programmer, you may make use of an extensive application programming interface (API). This API may be used remotely (meaning that your program can query a public server where the SeqHound database is maintained for you). As an example, you can write a program that will return a list of all proteins from a specific organism that have a known 3D structure and that have been annotated by GO as kinases. There are over 100 function calls available in PERL, Java, C and C++.
2. As a programmer, you may choose to use the local SeqHound API instead of the remote API. This will provide you with faster, private access but it also means that you have to install SeqHound locally on your own machine and maintain it yourself.
3. As a web user, you may access sequence records by searching for sequence record identifiers on the web-interface. SeqHound was primarily developed as a resource for programmers; as such the web-interface is very simple and represents only a limited number of functions available in SeqHound API. We are currently developing a web-interface that will allow a non-programmer to access many of the powerful functions provided by SeqHound.

To get started with using SeqHound, go to [\*section 2\*](#).



**How to contact us.**

General enquiries or comments can be posted to the SeqHound usergroup mailing list *seqhound.usergroup@blueprint.org*. You may also subscribe to this list to receive regular updates about SeqHound developments.

Private enquiries may be made to *seqhound@blueprint.org*.

Bug reports from external users should be reported to *seqhound@blueprint.org*.

Any errors or questions about this manual should be addressed to *seqhound@blueprint.org*.

## Who is SeqHound?

Chronologically ordered according to when the person first started work on SeqHound.

Chris Hogue  
Katerina Michalickova  
Gary Bader  
Ian Donaldson  
Ruth Isserlin  
Michel Dumontier  
Hao Lieu  
Marc Dumontier  
Doron Betel  
Renan Caverio  
Ivy Lu  
Rong Yao  
Volodya Grytsan  
Zhe Wang  
Victor Gu  
Rachel Farrall  
Michael Matan  
Elizabeth Burgess  
Kai Zheng

## About this manual.

This manual contains everything that has been documented about SeqHound.

If you can't find the answer here then please contact us. This manual was written and reviewed by the persons listed under "Who is SeqHound". Any errors should be reported to [seqhound@blueprint.org](mailto:seqhound@blueprint.org).

You can find out more about the general architecture of SeqHound by reading the *SeqHound paper* that is freely available from BioMed Central. This paper is included in the supplementary material distributed with this manual. See:

Michalickova K, Bader GD, Dumontier M, Lieu H, Betel D, Isserlin R, Hogue CW.  
*SeqHound: biological sequence and structure database as a platform for bioinformatics research.* **BMC Bioinformatics.** 2002 Oct 25;3(1):32.  
PMID: 12401134

Section 2 is a one page description that tells you what to read first to get started depending on what kind of user you are.

Section 3 is of interest to programmers who want to use the remote API to access information in the SeqHound database maintained by the Blueprint Initiative.

Section 4 is of interest to programmers and system administrators who want to set up SeqHound themselves so they can use the local API.

Section 5 is an in-depth description of everything that's in the SeqHound database and how it gets there (table by table). This section will be of interest to all users.

Section 6 describes how programmers can add to SeqHound. This section also describes our internal development process at Blueprint.

Section 7 includes Appendices of background and reference material.

**Conventions**

The following section describes the conventions used in this manual.

*Italic*

is used for filenames, file extensions, URLs, and email addresses.

`Constant Width`

is used for code examples, function names and system output.

**Constant Bold**

is used in examples for user input.

***Constant Italic***

is used in examples to show variables for which a context-specific substitution should be made.

## 2. How to get started

There are three ways to make use of SeqHound. How you intend to use it will determine which sections of this document you will want to read.

### **I'm a programmer and I want to use the remote API.**

You may make use of an extensive application programming interface (API). This API may be used remotely (meaning that your program can query a public server where the SeqHound database is maintained for you). As an example, you can write a program that will return a list of all proteins from a specific organism that have a known 3D structure and that have been annotated by GO as kinases. There are over 150 function calls available in PERL, Java, C and C++. SeqHound was designed primarily to support this remote API access. To get started with using the remote API, see [section 3](#). You may also want to use [section 5](#) as a reference for background details on the parts of SeqHound that you are using

### **I'm a programmer with system administrator skills and I want to install SeqHound locally.**

You may choose to use the local SeqHound API instead of the remote API. This will provide you with faster, private access but it also means that you have to install SeqHound locally on your own machine and maintain it yourself. Before attempting this you should first be familiar with the remote API and know that it meets your needs. To get started with installing a local version of SeqHound, see [section 4](#). You may also want to make use of [section 5](#) where the SeqHound system is described in detail.

### **I have internet access and I want to take a look at the contents of SeqHound.**

As a web user, you may access sequence records by searching for sequence record identifiers on the web-interface. SeqHound was primarily developed as a resource for programmers; as such the web-interface is very simple and represents only a limited number of the functions available in the SeqHound API. We are currently developing a web-interface that will allow a non-programmer to access many of the powerful functions provided by SeqHound. To get started using the web-interface, go to <http://seqhound.blueprint.org> and click on the *Seqhound WEB interface help* link.

### **I'm a programmer and I want to help develop SeqHound.**

SeqHound is developed as an open source project by the members of the Blueprint Initiative at the Samuel Lunenfeld Research Institute. These developers have read this manual in its entirety and (in many cases) written parts of it. The entire source code is posted on SourceForge at <http://sourceforge.net/projects/slrtools/>.

External developers who are interested in developing SeqHound should contact [seqhound@blueprint.org](mailto:seqhound@blueprint.org). [Section 6](#) of this manual contains background material of particular interest to SeqHound developers.

### 3. Using the SeqHound remote API

#### How to get started with the remote API.

To get started with using the remote API, follow the outline steps below and refer to the listed sections.

1. Define the problem you are trying to solve and determine whether using the SeqHound API can help you solve it.

To help with this, we have compiled a brief outline of what the SeqHound API allows you to do (see "[\*An overview of the SeqHound API\*](#)"). This two-page section will help you narrow down the set of functions that might be most useful to you (there are over 140 functions in the SeqHound API). For example, you might be most interested in functions listed under "[\*Complete Genome Iterators\*](#)".

This overview section is also hyperlinked to detailed descriptions for each of the individual functions. This detailed description of the API is maintained at [http://www.blueprint.org/seqhound/api\\_help/apifunctslist.html](http://www.blueprint.org/seqhound/api_help/apifunctslist.html).

2. Test the functions you want to use.

If the functions you have chosen return simple strings and/or integers, you can test them without setting up a program.

Refer to the list of underlying http calls for each API function at [http://www.blueprint.org/seqhound/api\\_help/httpcalls.html](http://www.blueprint.org/seqhound/api_help/httpcalls.html).

If, for example, you want to use the API function called "SHoundAccFromGi", you would use the call:

```
http://seqhound.blueprint.org/cgi-bin/seqrem?fnct=SeqHoundAccFromGi&gi=value
```

where "**value**" is replaced by a GenInfo (GI) identifier value like "6322454".

Also note that "**ShoundAccFromGi**" becomes "**SeqHoundAccFromGi**" in the http call.

Pasting this call into the url address bar of an internet browser you should see:

```
SEQHOUND_OK NP_012528
```

if everything is working correctly.

---

---

As a word of caution, you should be careful when using API calls that take lists as input. The output may not be in an order that corresponds with the order of input values. If order is important in your program, it is better to use a series of calls to a version of the function that takes only a single input value (not a list). For example you might want to use SHoundFindAcc rather than ShoundFindAccList.

Keyed list functions will be available shortly which will return an ordered list.

---

---

3. Decide which language to use.

The remote API is currently available in 4 languages: C, C++, Java and PERL.

In general, the majority of functions are supported by all four API's, however; only C and C++ will support functions that return NCBI data structures such as bioseqs and biostrucs. You should use the section below "*Language support for the API functions*" and the detailed API function descriptions listed at [http://www.blueprint.org/seqhound/api\\_help/apifunctsdet.html](http://www.blueprint.org/seqhound/api_help/apifunctsdet.html) to help you decide which language to use.

The remote PERL API is the easiest to set up. Setting up to use the C or C++ remote API is more difficult because it requires installing the NCBI C or C++ toolkit; however, detailed instructions are provided for doing this in this manual.

4. Follow the instructions to set up the API development environment for your language of choice.

*Using the remote API for C (Unix)*

*Using the remote API for C++ (Unix).*

*Using the remote API for Java.*

*Using the remote API for PERL.*

Detailed instructions are included for each language along with example programs.

## An overview of the SeqHound API.

This section describes the types of functions available in the SeqHound Application Programming Interface (API). The functions are sorted into groups of related functions. Click on any one of the functions in a group for a more detailed description.

### gi centric

A GenBank GenInfo (GI) identifier is the primary identifier used by GenBank to uniquely identify sequence records. The SeqHound API is GI-centric in that many functions use this identifier as a key to retrieve sequence and sequence annotation data. Identifiers from other databases may be converted to GI's using the API call "ShoundFindName" or "SHoundGiFromPDBChain" for PDB identifiers.

### lists

Many API functions have two forms (one that takes a single query as input and one that takes a list of queries). Programmers should be careful when using API calls that take lists as input. The output may not be in an order that corresponds with the order of input value since each input value may return no result or may return an unpredictable number of results. If order is important in your program, it is better to use a series of calls to a version of the function that takes only a single input value (not a list) or use the more recent keyed list functions available for some of the list functions.. For example you might want to use SHoundFindAcc rather than ShoundFindAccList or SHoundGetFastaKeyedList instead of SHoundGetFastaList. More keyed list functions will be available in future releases.

### data structures in asn.1 and xml

Some API functions return data structures in ASN.1 binary format. These structures are either defined by the NCBI (like, bioseqs and biostrucs) or by the SLRI (like the FlinkSet).

More information on NCBI data structure can be found by going to <http://www.ncbi.nlm.nih.gov/IEB/ToolBox/SB/hbr.html> , entering the name of the structure into the query box and checking the box beside "asns" before hitting the "Submit query button".

More information on SLRI data structures can be found at:

<http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slritools/slri/seqhound/asn>

For programmers who do not wish to use ASN.1, many of these functions have alternate forms that return a data structure as an XML formatted string.



This list last updated March 24, 2004

## SYSTEM INITIALIZATION, FINISHING AND TESTING

Functions in this group will be used by all SeqHound API programs to open a connection to SeqHound at the beginning of the program and to close the connection at the end of the program.

SHoundFini

SHoundInit

SHoundIsInited

SHoundIsNetEntrezOn

SHoundNetEntrezInit

## GENBANK ID CONVERSIONS

GenBank sequence records are associated with three types of identifiers; Gene Info identifiers (GI), GenBank accessions and other identifiers kept in the record's "seq-id". These functions may be used to convert between these three types of identifiers. While most people are familiar with the Genbank accession and GI identifiers, GenBank may also keep identifiers that denote the source database and identifier used by GenBank to construct the record (for example embl or pir specific accessions or Blattner identifiers (BLATT)). These identifiers are usually not displayed in the GenBank flat-file but are present in the underlying ASN.1 of the record in the seq-id structure. You can read more about this structure by going to

<http://www.ncbi.nlm.nih.gov/IEB/ToolBox/SB/hbr.html> and entering "seq-id" in the query box or by searching for "seq { id {" in the ASN.1 format of a GenBank record. A complete list of the dbnames and example identifiers that may be used with these functions is listed in the API supplementary information page. It should be noted that there is a conceptual difference between functions in this section and those in the DBXref module. Conversion between GI's, Accessions and "other" database identifiers are conversions between identifiers that point to exactly the same sequence record. In contrast, database cross-references found in source records (see DbXref module) may refer to other things (for example, annotations on or records related to the source record).

SHoundDbNameAndIdListFromGBAcc  
SHoundDbNameAndIdListFromGBAccKeyedList  
SHoundDbNameAndIdListFromGi  
SHoundDbNameAndIdListFromGiKeyedList  
SHoundGBAccFromDbNameAndId  
SHoundGBAccFromDbNameAndIdKeyedList  
SHoundGBAccFromGi  
SHoundGBAccFromGiKeyedList  
SHoundGetAccFromRetiredGi  
SHoundGiFromDbNameAndId  
SHoundGiFromDbNameAndIdKeyedList  
SHoundGiFromGBAcc  
SHoundGiFromGBAccKeyedList  
SHoundSeqIdFromGi  
SHoundSeqIdFromGiList  
SHoundTitleFromGi

## SEQUENCE AND STRUCTURE CHECKS

These functions may be used to do things like tell whether a given GenInfo identifier (GI) refers to a protein sequence record or whether it exists in the SeqHound database.

SHound3DbWhat  
SHound3DExists  
SHoundExists  
SHoundGetLargestMMDBID  
SHoundIsNRFirst  
SHoundIsProtein  
SHoundMoleculeType  
SHoundSequenceLength

## SEQUENCE FETCH ASN.1 BIOSEQ

These functions return a sequence record as a "Bioseq". A "Bioseq" is an NCBI data structure used to store sequence data. To see a complete description of a Bioseq in ASN.1 format, go to <http://www.ncbi.nlm.nih.gov/IEB/ToolBox/SB/hbr.html> , type in the query "Bioseq" check the box beside "asns" and hit the "Submit query button".

SHoundGetBioseq

SHoundGetBioseqList

SHoundGetXMLBioseq

## SEQUENCE FETCH ASN.1 BIOSEQSET

These functions return a set of sequence records as a bioseqset (also see functions under "SEQUENCE FETCH ASN.1 BIOSEQ").

SHoundGetBioseqSet

SHoundGetBioseqSetList

## SEQUENCE FETCH ASN.1 SEQENTRY

A Seq-entry is an NCBI data structure used to hold Bioseqs and Bioseq-sets (see above). To see a complete description of a bioseq in ASN.1 format, go to <http://www.ncbi.nlm.nih.gov/IEB/ToolBox/SB/hbr.html> , type in the query "Seq-entry", check the box beside "asns" and hit the "Submit query button".

SHoundGetSeqEntry

SHoundGetSeqEntryList

SHoundGetXMLSeqEntry

## SEQUENCE FETCH FASTA

These functions return a sequence record(s) in FASTA format given a GenBank GenInfo identifier (GI).

SHoundGetFasta

SHoundGetFastaFromRedundantGroupID

SHoundGetFastaFromRedundantGroupIDKeyedList

SHoundGetFastaKeyedList

SHoundGetFastaList

## DEFLINE FETCH

These functions return a definition line for a sequence record(s) given a GenBank GenInfo identifier (GI).

SHoundGetDefline

SHoundGetDeflineList

## SEQUENCE FETCH GENBANK FLAT FILE

These functions return a sequence record(s) in GenBank flat file format given a GenBank GenInfo identifier (GI).

SHoundGetGenBankff

SHoundGetGenBankffList

SHoundGetSeqFromGenBank

## STRUCTURE FETCH

These functions return a biological structure record in either the Molecular Modeling Database (MMDB) format or the Protein Database (PDB) format.

SHoundGet3D

SHoundGet3DEx  
SHoundGet3DfromPdbId  
SHoundGet3DfromPdbIdEx  
SHoundGetPDB3D  
SHoundGetXML3D

## SEQHOUND LINKS

These functions return links to other databases found in a sequence or structure record. For instance, given the GI of a protein sequence record, you can retrieve links to the nucleic acid record that encodes that protein or the organism that the protein belongs to.

SHound3DFromGi  
SHound3DFromGiList  
SHoundDNAFromProtein  
SHoundDNAFromProteinList  
SHoundDNAFromTaxID  
SHoundDNAFromTaxIDIII  
SHoundDNAFromTaxIDList  
SHoundGetReferenceIDFromGi  
SHoundGetReferenceIDFromGiList  
SHoundGiFrom3D  
SHoundGiFrom3DList  
SHoundGiFromPDBchain  
SHoundGiFromReferenceID  
SHoundGiFromReferenceList  
SHoundMuidFrom3D  
SHoundMuidFrom3DList  
SHoundMuidFromGi  
SHoundMuidFromGiList  
SHoundProteinFromDNA  
SHoundProteinFromDNAList  
SHoundProteinsFromTaxID  
SHoundProteinsFromTaxIDIII

SHoundProteinsFromTaxIDList  
SHoundTaxIDFrom3D  
SHoundTaxIDFrom3DList  
SHoundTaxIDFromGi  
SHoundTaxIDFromGiList

## COMPLETE GENOME ITERATORS

These functions returns lists of things that pertain to completely sequenced genomes. For example, given a taxon identifier, these functions can retrieve a complete list of protein, nucleic acid or chromosome records associated with the completely sequenced genome for that organism.

SHoundAllGenomes  
SHoundChromosomeFromGenome  
SHoundChromosomeFromGenomeList  
SHoundDNAFromChromosome  
SHoundDNAFromChromosomeList  
SHoundDNAFromOrganism  
SHoundDNAFromOrganismList  
SHoundProteinsFromChromosome  
SHoundProteinsFromChromosomeList  
SHoundProteinsFromOrganism  
SHoundProteinsFromOrganismList

## REDUNDANT (EQUIVALENT) SEQUENCES

Sequence records in GenBank are redundant. For example, two different sets of authors may submit the same sequence to GenBank. Both groups will be given different sequence record identifiers (GI's). SeqHound, groups sequences that are exactly the same under a single identifier (Redundant Group Identifier). These functions allow the programmer to find the Redundant Group that a sequence record (GI) belongs to and then retrieve all of the other sequences that belong to the same group. This set of functions is useful when trying

to collect all of the annotation that applies to a biological sequence but may be scattered across several sequence records.

SHoundFirstOfRedundantGroupFromID  
SHoundFirstOfRedundantGroupFromIDList  
SHoundRedundantGroup  
SHoundRedundantGroupFromID  
SHoundRedundantGroupFromIDList  
SHoundRedundantGroupIDFromGI  
SHoundRedundantGroupIDFromGIList  
SHoundRedundantGroupKeyedList  
SHoundRedundantGroupList

## TAXONOMY

This set of functions allows the programmer to traverse the NCBI taxonomy tree.

SHoundGetAllTaxAncestors  
SHoundGetAllTaxProgeny  
SHoundGetTaxChildNodes  
SHoundGetTaxChildNodesList  
SHoundGetTaxLineageFromTaxID  
SHoundGetTaxNameFromTaxID  
SHoundGetTaxNameFromTaxIDByClass  
SHoundGetTaxParent  
SHoundIsTaxDeleted  
SHoundIsTaxMerged

## SEQUENCE NEIGHBOURS

This set of functions retrieves sequence neighbours and sequence alignments given a GenInfo (GI) identifier of a protein. The non-redundant set of protein sequence records in SeqHound are used to precompute neighbours and alignments

that are stored by SeqHound. This process is described in the NBLAST paper available in BioMed Central.

SHound3DNeighboursFromGi  
SHound3DNeighboursFromGiEx  
SHound3DNeighboursFromGiList  
SHound3DNeighboursFromTaxID  
SHoundGetBlastResult  
SHoundGetBlastSeqAlign  
SHoundGiAndNumNeighboursList  
SHoundNeighboursFromGi  
SHoundNeighboursFromGiEx  
SHoundNeighboursFromGiList  
SHoundNeighboursFromTaxID  
SHoundNeighboursOfNeighbours  
SHoundNeighboursOfNeighboursList  
SHoundNumNeighboursInDB

## COMPLETE GENOME ANALYSIS

These functions return information concerning completely sequenced genomes that are present in SeqHound.

SHound3DListFromTaxID  
SHoundAllGenomesWithNumProteins  
SHoundAllGenomesWithScientificNames

## FUNCTIONAL ANNOTATION

These functions retrieve functional annotation related to sequence records in SeqHound. This annotation includes associated identifiers from the Gene Ontology (GO), Online Inheritance in Man (OMIM), NCBI's Conserved Domain Database (CDD) and LocusLink.

SHoundCDDIDFromGi  
SHoundCDDIDFromGiList



SHoundCDDIDFromLLID  
SHoundCDDScoreFromGi  
SHoundGiFromCDDID  
SHoundGiFromCDDIDList  
SHoundGiFromGOID  
SHoundGiFromGOIDAndECode  
SHoundGiFromGOIDList  
SHoundGiFromGOIDListAndECode  
SHoundGiFromLLID  
SHoundGiFromLLIDList  
SHoundGiFromOMIM  
SHoundGiFromOMIMList  
SHoundGOAnnotFromRedundantGi  
SHoundGOECodeFromGiAndGOID  
SHoundGOIDFromGi  
SHoundGOIDFromGiList  
SHoundGOIDFromLLID  
SHoundGOIDFromRedundantGi  
SHoundGOIDFromRedundantGiList  
SHoundGOPMIDFromGiAndGOID  
SHoundLLIDFromCDDID  
SHoundLLIDFromGi  
SHoundLLIDFromGiList  
SHoundLLIDFromGOIDAndECode  
SHoundLLIDFromOMIM  
SHoundLocusFromGi  
SHoundOMIMFromGi  
SHoundOMIMFromGiList  
SHoundOMIMFromLLID  
SHoundProteinReportSetDescription

## GO HIERARCHY

These functions allow the programmer to traverse the Gene Ontology directed acyclic graph. See <http://www.geneontology.org/> for more information on this data structure.

SHoundGODBGetAllAncestors  
SHoundGODBGetAllChildren  
SHoundGODBGetChildrenOf  
SHoundGODBGetClassification  
SHoundGODBGetNameByID  
SHoundGODBGetParentOf  
SHoundGODBGetRecordByReference

## RPS BLAST DOMAINS

This group of functions is used to correlate conserved domains with protein sequence records. Conserved domains are precomputed for a non-redundant set of proteins in SeqHound using RPS BLAST and the Conserved Domain Database (CDD) from the NCBI.

SHoundGetDomainIdFromLabel  
SHoundGetDomainLabelFromDomainId  
SHoundGetDomainsFromGi  
SHoundGetDomainsFromGiListWithEvalvalue  
SHoundGetDomainsFromGiWithEvalvalue  
SHoundGetFilteredDomainsFromGiWithEvalvalue  
SHoundGetGisByDomainId  
SHoundGetGisByDomainIdAndEvalvalue  
SHoundGetGisByNumberOfDomains

## DEPRECATED FUNCTIONS

The functions listed in this section have been deprecated and may be replaced by new functions that may have different parameters and/or return types. Functions are listed here for historical purposes only. The description of each

function may list a comparable new function that can be used instead. While deprecated functions will be supported, developers should use the new function listed going forward if at all possible. If the function has not been written for your language of choice (see the function tracker at <http://www.blueprint.org/seqhound/apifunctsstatus.html>), you may request that the function be written by sending an email to [seqhound@blueprint.org](mailto:seqhound@blueprint.org). This will help us to direct our development efforts. Once the versions of the new function (that replaces a deprecated function) have been written into the remote Java, Perl, C and C++ remote API's we may consider removing support for these calls (with ample warning and in consideration of feedback). At this time, these functions would be moved to a section entitled Unsupported Functions.

SHoundAccFromGi

SHoundAccFromGiList

SHoundFindAcc

SHoundFindAccList

SHoundFindName

SHoundFindNameList

SHoundGetNameByGi

## Underlying HTTP calls.

A complete list of all of the http calls that support the SeqHound API can be found at:

*[http://www.blueprint.org/seqhound/api\\_help/httpcalls.html](http://www.blueprint.org/seqhound/api_help/httpcalls.html)*

The list is maintained at that location.

The SeqHound remote API is based on hyper-text transfer protocol (http) calls. Each API function has a corresponding call. You can test any one of the SeqHound API functions using any internet browser.

For example, if you want to test the API function called “SHoundAccFromGi”, you would use the call:

**`http://seqhound.blueprint.org/cgi-bin/seqrem?fnct=SeqHoundAccFromGi&gi=value`**

where “***value***” is replaced by a GenInfo (GI) identifier value like “6322454”.

Also note that “**Shound**AccFromGi” becomes “**SeqHound**AccFromGi” in the http call.

Pasting this call into the url address bar of an internet browser you should see:

`SEQHOUND_OK NP_012528`

if everything is working correctly.

**The complete SeqHound API description.**

All of the functions in the SeqHound API are described on a single web-page located at:

*[http://www.blueprint.org/seqhound/api\\_help/apifunctsdet.html](http://www.blueprint.org/seqhound/api_help/apifunctsdet.html)*

The API description is maintained at that location.

**Language support for API functions.**

Function status and language availability are available on

*[http://www.blueprint.org/seqhound/api\\_help/apifunctsstatus.html](http://www.blueprint.org/seqhound/api_help/apifunctsstatus.html)*

## Using the remote API for C

Currently, the SeqHound development team distributes precompiled C libraries for the Linux Intel-x86 (32 bit) and Sun Microsystem Solaris sparc (64 bit) platforms. Developers writing C programs on these platforms can take advantage of the precompiled libraries without having to build the libraries themselves. Programmers writing C programs on other platforms will need to compile the libraries as described in Section 4.4.

The following steps are needed to make remote SeqHound API calls:

- 1) Specify the home path for user "seqhound" (e.g. export SEQHOUND\_HOME=/home/seqhound).
- 2) Download the binary distribution from the Blueprint FTP site (e.g. [ftp://ftp.blueprint.org/pub/SeqHound/Execs/seqhoundx.xx.linux\\_x86\\_32bin.tar.gz](ftp://ftp.blueprint.org/pub/SeqHound/Execs/seqhoundx.xx.linux_x86_32bin.tar.gz)).
- 3) Unpackage the distribution in the home directory of user "seqhound" (e.g. `tar -xzf seqhoundx.xx.linux_x86_32bin.gz`).
- 4) In the directory /home/seqhound/example, run the shell script `makeexample.sh` (`./makeexample.sh`) to build the executable `example_remapi`.
- 5) The default setting in the configuration file `.shoundremrc` points to the http server at `seqhound.blueprint.org`. One can invoke the example program by typing `./example_remapi` in the example directory. The program will query the cgi on the seqhound http server and display results to the standard output. The source file `example.c` can be modified to test other SeqHound API calls. SeqHound test driver `shunittest.c` contains a list of SeqHound APIs. The test driver source code is part of the SeqHound source code distribution, which can be downloaded from the Blueprint FTP site([ftp://ftp.blueprint.org/pub/SeqHound/Code/seqhound\\_x.x\\_src.tar.gz](ftp://ftp.blueprint.org/pub/SeqHound/Code/seqhound_x.x_src.tar.gz))

## Using the remote API for C++\*

---

Note: The following section is under revision and will be updated shortly.  
Please contact SeqHound at *[seqhound@blueprint.org](mailto:seqhound@blueprint.org)* for the latest update.

---



## Using the remote API for Java

Last updated March 16, 2005

FIRST TIME USERS OF THE SEQHOUND JAVA API MAY SKIP ‘SPECIAL NOTES’ AND GO DIRECTLY TO THE SYSTEM REQUIREMENTS SECTION.

IF YOU ARE CURRENTLY USING THE 3.01 VERSION (OR EARLIER VERSION) OF THE SEQHOUND JAVA API, DO NOT INSTALL LATER VERSIONS UNTIL YOU HAVE READ THE ‘SPECIAL NOTES’ SECTION BELOW. THIS SECTION DESCRIBES SMALL BUT CRUCIAL CHANGES THAT MUST BE MADE TO REMOTE APPLICATIONS USING THE SEQHOUND JAVA API (v 3.2 AND LATER). APPLICATIONS USING THE SEQHOUND JAVA API (v3.01 AND EARLIER) WILL CONTINUE TO BE SUPPORTED, HOWEVER, DEVELOPERS ARE STRONGLY ENCOURAGED TO TAKE THE TIME TO UPGRADE TO THE LATEST VERSION.

### Special Notes

The seqhound java API framework has been significantly refactored for release 3.2 and later. This refactoring was done in concert with the development of a Java API implementation which directly accesses Seqhound database tables, rather than having to go through a remote API server first, as well as the development of a Java based remote API server. Due to this refactoring, many characteristics of how to use the java API have changed, though, in general, all the functionality found in previous releases of the Java API are still present. This section details these changes to help users of the previous API efficiently modify their code to utilize the new API.

#### *Major Changes:*

- \* API implementation which utilizes direct local DB access to Seqhound tables. The new Seqhound API functions in this release are available with both local and remote access implementations; i.e., implementations for these functions are available which query a remote seqhound server (what you are used to) or which can query a seqhound database server directly (if you set one up locally). To learn how to configure a seqhound client to access a local seqhound database instance, see the section on configuring the seqhound Java API below.
- \* Retirement of class `org.blueprint.seqhound.SeqHound`. This class was the Seqhound remote API client in previous releases. This class has been moved to the location `org.blueprint.seqhound.queries.seqrem.SeqHound`, and can be accessed there directly if absolutely necessary, though it is recommended that users migrate to using the new framework, described in a later section. This **class** has been replaced with the **interface** ‘`org.blueprint.seqhound.Seqhound`’, and objects which implement that interface.

- \* API definition changes. The seqhound java API is now defined by the interface class `org.blueprint.seqhound.Seqhound`. This interface contains definitions for the methods defined by the Seqhound java API of previous releases, with the following changes:
  - \* All methods throw exceptions of types `SeqhoundException` and `SeqhoundLogicException`, and no other exceptions. This follows the Convert Exceptions design pattern, which specifies that methods should only throw exceptions belonging to the problem domain which they belong to. Any lower level exception thrown by interface method implementations, such as network, file or db access exceptions, will be wrapped with seqhound exceptions before being thrown to clients.
- 3.2) The return types of methods `SHoundGetFastaFromRedundantGroupIDKeyedList` and `SHoundGetFastaKeyedList` have been changed from `HashTable` to `ShRIDFastaTable` and `ShGiFastaTable`, respectively. This change was made in order to more strongly type the return types, so that they would be more compatible with the new remote API framework developed for this release. These new return types are subtypes of `HashTable`, so this change should not, in general, break backwards compatibility.
- 3.3) The method `SHoundGetFastaList(int[] giList, Writer out)` has been renamed `ShoundGetFastaListToWriter`

4.) Seqhound client object construction. The creation of Objects implementing the Seqhound API interface has been abstracted behind an interface provided by the class `org.blueprint.seqhound.SeqhoundFactory`. This change was necessitated because the new framework allows the mixing of local and remote implementations of methods within the same Seqhound interface implementation. The complexity of creating objects with mixed implementations, which involves Dynamic Proxy Objects, was deemed to be better hidden behind the interface of a Factory class. The `SeqhoundFactory` class provides various `createSeqhound` methods for creating implementations of the Seqhound interface which implement methods using either the remote or local interfaces, as defined in a configuration file. Classes implementing the Seqhound interface cannot be directly instantiated. Whereas in previous releases, one would instantiate a seqhound client using the following code:

```
SeqHound sh = new SeqHound();
```

In the new framework, an application would create an instance of a Seqhound client using the following code:

```
SeqhoundFactory shf = new SeqhoundFactory()  
Seqhound sh = shf.createSeqhound();
```

or, alternatively, using a non-default `SeqHoundProperties` object to configure with:

```
SeqhoundFactory shf = new SeqhoundFactory()  
SeqhoundProperties shp = new SeqhoundProperties();  
//.. set various properties of shp  
Seqhound sh = shf.createSeqhound(shp);
```

5.) The *.shoundremrc* configuration file has been replaced with the *shconfig.properties* configuration file. See the section below on remote client configuration for description of it's format

6) New API functions for converting between GenBank Identifiers. 12 new functions have been added to seqhound as part of the GenBank ID Conversions API module. These functions rely on the Seqhound core sequence tables(specifically accdb), and in many cases serve to replace deprecated sequence conversion functions. See the API description page at <http://www.blueprint.org/seqhound/apifunctslist.html> and the Function tracker at <http://www.blueprint.org/seqhound/apifunctsstatus.html>.

7. Release of the jseqrem web application servlet. This web application is a java complement of the C seqrem CGI which serves most remote API clients. All new remote API functions are served by jseqrem. This application's main servlet class is `org.blueprint.seqhound.jseqrem.Jseqrem`. The URL of the public jseqrem is <http://seqhound.blueprint.org/seqhound/jseqrem>

## System Requirements

In order to build and use SeqHound Java on your computer you will need the Java 2 Runtime Environment installed, version 1.4 or later. See <http://www.java.sun.com/j2se> for further details.

SeqHound comes with Ant build scripts to ease the task of compiling and building libraries from the source code distribution, so it is recommended that you also have Ant installed.

Additionally, the source code tree doubles as a project folder for the Eclipse IDE, so developers may also wish to use Eclipse for java development.

Ant is available from <http://ant.apache.org/>. Eclipse is available from: <http://www.eclipse.org/>

## Using the pre-compiled binary distribution.

You can download the seqhound remote and local binary jars and use them immediately in your program. These are distributed at <ftp://ftp.blueprint.org/pub/SeqHound/Code/seqhound-java-x.x.bin.tar.gz>.

If you use this distribution you may follow the instructions below and then skip the next section (Compile, Build and Install Java) and go directly to “Using the SeqHound Java API”.

### *Binary Distribution Archive Contents*

This archive contains only what a developer would need to get started using the Seqhound Java library:

- a.) *seqhound-java-x.x.jar* (jar archive containing the compile class files which make up the seqhound java library version x.x)
- b.) *seqhound-java-remote-x.x.jar* (jar archive containing only those compiled class files of the seqhound java library required for remote API clients)
- c.) *seqhound-java-tests-x.x.jar* (jar archive containing the compile class files which make up the seqhound test classes)
- d.) *RELEASE* (release notes for this version)
- e.) *README*
- f.) *doc/* (directory containing the javadoc documentation for the distribution)
- g.) *lib/* (directory containing the 3<sup>rd</sup> party library dependancies which seqhound's java client depends upon, in jar archive format)
- h.) *shconfig.properties* ( sample seqhound client configuration file. The seqhound client library uses this to configure logging, database connection and/or remote seqhound connections. **This default file will work just fine for most users. If you are using a local installation of SeqHound, you must read the manual section on “Configuring Data Sources”. )**

Here's an example set up:

```
tar xvf seqhound-java-X.X-bin.tar.gz
```

```
cd seqhound-java-X.X-bin
```

**Test and verify your binary SeqHound Java API installation.**

Included with the seqhound library are several classes for verifying that your seqhound build is working properly.

These test classes were built using the junit framework, and iterate through nearly all seqhound API functions to verify that they are working correctly in your configuration. These classes are `org.blueprint.seqhound.SeqhoundAPITest` and `org.blueprint.seqhound.SeqhoundTest`. Each performs tests on a separate subset of the seqhound API, so they are complementary and should both be run to verify your configuration (these tests may be consolidated into a single class in a future release).

If you have ant installed, you may execute these tests by entering:

```
ant test
```

The ant test has the benefit that it generates a junit report in a browseable html format. The results of the junit test are output to the directory `$seqhound_java_home/test_results`, and you may view the html formatted report by loading the `$seqhound_java_home/test_results/index.html` in a web browser.

If you don't have ant installed you can still run the tests by entering the following commands while in the `seqhound-java-X.X-bin/` directory:

```
export CLASSPATH=lib/commons-httpclient-2.0.2.jar:lib/commons-logging.jar:lib/junit.jar:seqhound-java-3.2-bin.jar:seqhound-java-tests-3.2-bin.jar:lib/log4j-1.2.8.jar:lib/blueprint-commons-db-0.2.jar:lib/commons-pool-1.2.jar:lib/commons-collections-3.1.jar:lib/mysql-connector-java-3.0.14-production-bin.jar:lib/commons-dbcp-1.2.1.jar
```

```
java org.blueprint.seqhound.SeqhoundAPITest
```

```
java org.blueprint.seqhound.SeqhoundTest
```

When you have successfully completed this step, go to the section "Using the SeqHound Java API" below.

## Compile, Build and Install Seqhound Java

Alternatively, if you wish to compile SeqHound yourself, download the source tar archive seqhound-java-X.X-src.tar.gz , distributed at

<ftp://ftp.blueprint.org/pub/SeqHound/Code/seqhound-java-x.x-src.tar.gz>.

and follow these steps:

```
tar xvf seqhound-java-X.X-src.tar.gz
cd seqhound-java-X.X
#to compile the source code
ant compile
#to create a jar containing all the seqhound local and
#remote client library class files
ant dist-local

or

#to create a jar containing only those class files
required for the seqhound remote distribution.

ant dist-remote
```

### *Source Distribution Archive Contents*

The tar archive of the seqhound source tree is in the form of an Eclipse(™) project directory. This may be imported directly into an Eclipse workspace and developed with. Alternatively, users can use the included ant build.xml and it's associated targets (described below) to perform the tasks of building the library and running the tests.

- a.) *java/src* directory ( the source code tree for the seqhound library classes )
- b.) *java/test* directory ( the source code tree for the test classes )
- c.) *doc/* directory ( the javadoc documentation for the source code tree )
- d.) *dist/* directory ( where compiled jar libraries are placed )
- e.) *bin* directory ( where compiled binary class files are placed )
- f.) *conf/* directory ( configuration files used in building of different ant targets )
- g.) *lib/* directory ( third party jar libraries which seqhound is dependant upon )
- h.) *.classpath* file ( Eclipse project classpath file )
- i.) *.project* file ( Eclipse project file )



- j.) *shconfig.properties* file ( sample seqhound client configuration file. The seqhound client library uses this to configure logging, database connection and/or remote seqhound connections. See the manual section on configuration for more details )
- k.) *RELEASE* (release notes for this version)
- l.) *README*
- m.) *build.xml* file (ant build file)

The ant build file contains targets for various seqhound development tasks, such as compiling the source, building library jars and creating javadoc documentation. The available targets are:

- `compile` - compile all source files and place them in the `/bin` directory, preserving classpath/directory structure
- `dist-remote` - builds a jar of the seqhound class files required for running a remote Seqhound client, placing it in `dist/lib/`
- `dist-local` - builds a jar containing all seqhound class files, placing it in `dist/lib/`
- `dist-war` - builds a Web Application aRchive (WAR) file for the jseqrem seqhound remote servlet, which can be loaded into a java web application server (eg Apache's Jakarta-Tomcat ), placing it in `dist/war/`
- `tomcat-deploy` - deploys the (WAR) file built by the `dist-war` target to a running tomcat server, as specified in the `conf/tomcat.properties` file
- `tomcat-undeploy` - undeploys the web application deployed by the `tomcat-deploy` target
- `tomcat-redeploy` - undeploys and redeploys the jseqrem servlet web application; synonymous with calling 'tomcat-undeploy' and 'tomcat-deploy' in sequence.
- `test` - runs the junit tests for `org.blueprnt.seqhound.SeqhoundAPITest` and `org.blueprint.seqhound.SeqHoundTest`, writing results to the `test_results` directory in html format
- `clean` - cleans the project by erasing any compiled binaries, jars, documentation, and test results

Note that to use the tomcat related tasks, one will need to have the tomcat jar archive `catalina-ant.jar`, which defines those tasks, in your classpath. Similarly, one will need to have the `junit.jar` archive (found in the `lib` directory) in ones classpath to use the ant target 'test'.

## Testing and verifying your compiled source installation of the SeqHound Java API.

Included with the seqhound library are several classes for verifying that your seqhound build is working properly.

These test classes were built using the junit framework, and iterate through nearly all seqhound API functions to verify that they are working correctly in your configuration. These classes are `org.blueprint.seqhound.SeqhoundAPITest` and `org.blueprint.seqhound.SeqhoundTest`. Each performs tests on a separate subset of the seqhound API, so they are complementary and should both be run to verify your configuration (these tests may be consolidated into a single class in a future release).

If you have ant installed, you may execute these tests by entering:

```
ant test
```

The ant test has the benefit that it generates a junit report in a browseable html format. The results of the junit test are output to the directory `$seqhound_java_home/test_results`, and you may view the html formatted report by loading the `$seqhound_java_home/test_results/index.html` in a web browser.

If you don't have ant installed, you can execute the test programs as normal java programs. You will need to have compiled the library and test cases to the 'bin' directory first. While in the `seqhound-java-x.x-src/` directory, enter on a single line:

```
java -classpath lib/commons-httpclient-2.0.2.jar:lib/commons-logging.jar:junit.jar:lib/log4j-1.2.8.jar:bin/  
org.blueprint.seqhound.SeqhoundAPITest
```

Then to run the second test, enter the following on a single line:

```
java -classpath lib/commons-httpclient-2.0.2.jar:lib/commons-logging.jar:lib/junit.jar:lib/log4j-1.2.8.jar:bin/  
org.blueprint.seqhound.SeqhoundTest
```

Note that the junit library jar, which is included with the tarball distributions of seqhound-java, must be on your classpath before you can use any of the above three options.

## Using the SeqHound Java API

A sample application is included with the seqhound distribution. This application takes GenBank GeneInfo identifier (GI) and retrieves the GenBank flat file summary of the protein or nucleic acid which corresponds to that GI. Note that you must have the necessary library JAR files included in your java classpath before you can execute the function; all necessary JAR dependancies should be included in the tarball distributions.

Example invokation:

```
java org.blueprint.seqhound.GetGenBankff 333
```

The source code for this function illustrates how to create seqhound client applications. The rest of this section will detail how to initialize and use the seqhound client libraries.

### *Seqhound Client Object Creation*

Seqhound client objects are objects which implement the Seqhound API defined by the interface `org.blueprint.seqhound.Seqhound`. These objects are manufactured by the `org.blueprint.seqhound.SeqhoundFactory` class, and their behaviour is determined by the values in the `org.blueprint.seqhound.SeqhoundProperties` object used by `SeqhoundFactory` in their creation. Objects of the `SeqhoundProperties` class contain the specification of what implementations to use for each method of the interface (local db or remote web-service) and how to access the datasources which the implementations depend upon. By default, `SeqhoundProperties` objects load the information contained in the `shconfig.properties` file in the current working directory when they are instantiated. They can also have their properties set programmatically, as `SeqhoundProperties` is a subclass of `java.util.Properties`. The average application will probably rely on the `shconfig.properties` file to determine client implementation details, and would use the following code to instantiate a seqhound client:

```
SeqhoundProperties shp = new SeqhoundProperties();  
SeqhoundFactory shf = new SeqhoundFactory();  
Seqhound sh = shf.createSeqhound(shp);
```

Note that the `SeqhoundProperties` constructor and the `createSeqhound` method throw `SeqhoundException` when they are unable to properly initialize the properties or the client, respectively. In some cases, they will emit warning messages when there are non-fatal issues with the seqhound client configuration.

### *Using Seqhound Client Objects*

Once instantiated, seqhound client objects can be used like any regular java object, and the methods of the Seqhound API can be invoked through it. Eg :

```
String gbff = sh.SHoundGetGenBankff(333);
```

All methods of the API throw SeqhoundException and SeqhoundLogicException. The former usually signals a seqhound system problem which the user may not be able to correct (such as a database server being down) whereas the later exception generally signals a problem in the configuration of seqhound which the user should be able to correct.

Developers may find more details on controlling the logging files in the 'Configuration Section' below.

### API Documentation

The SeqHound API is described in this manual in the section entitled 'An overview of the SeqHound API'. This section is a repeat of the page found at

<http://www.blueprint.org/seqhound/apifunctslist.html>

and contains links to more detailed descriptions of each API function found at

<http://www.blueprint.org/seqhound/apifunctsdet.html>.

These pages describe the API in general terms and list specifics for the C, C++ and Perl versions of the functions. Details that are specific to the Java implementations of the API functions are available at <http://www.blueprint.org/seqhound/javadocs/index.html>.

The main classes of interest are `org.blueprint.seqhound.Seqhound`, the Seqhound API definition, and `org.blueprint.seqhound.SeqhoundFactory`, the Factory class used to construct objects implementing the Seqhound API.

An html format of the JavaDocs are also available offline in the *docs/javadoc* directory of each tarball distribution. Open the file *index.html* in your favourite browser to view the documentation. You can also access the source code directly in the *java/src* directory.

## Configuration

In previous releases, the seqhound java client library depended upon the configuration file *.shoundremrc*. With this release, the configuration file which the seqhound client depends upon is the *shconfig.properties* file, which must be in the current working directory of the application using the library. This file is loaded into objects of the SeqhoundProperties class when they are instantiated with the default parameterless constructor. SeqhoundProperties objects are used in configuration of Seqhound interface implementations created by the SeqhoundFactory class. The *shconfig.properties* file is used to configure options such as what log4j logging level to use, what datasources to rely upon (such as direct database access and/or remote seqrem servers) and how to access those data sources.

Example *shconfig.properties* files can be found in the root directory of the binary and source tarball distributions. Alternatively, a default *shconfig.properties* file can be found on the public seqhound web server at:

<http://seqhound.blueprint.org/shconfig.properties>

This set of default configuration properties will configure your application to access the public seqhound web service servers to answer API calls, and log to a *shound.log* file.

### *shconfig.properties file*

*shconfig.properties* is in java properties file format. It contains 4 types of information:

- \* log4j settings, for configuring logging
- \* What datasources to use in the API implementations
- \* Method-specific data source overrides.
- \* Configuration on how to access the specified datasources (eg seqrem/db URLs)

The last 3 types of information are only of interest to users who have their own local instance of SeqHound.

### *1. Logging Configuration*

The *shconfig.properties* file is also used for configuring logging for the seqhound client. Seqhound java uses the log4j framework for logging messages. Logging is configured using the properties file configuration format for log4j. For example, to have the seqhound client log messages at the INFO level or higher to the file *shound.log*, enter the following lines in *shconfig.properties*:

```
#sets the rootLoggers logging level to info
log4j.rootLogger=INFO, Logfile
```

```
#specifies creation of a file appender, which will append log messages  
to the shound.log file
```

```
log4j.appender.Logfile=org.apache.log4j.FileAppender
```

```
log4j.appender.Logfile.File=shound.log
```

```
log4j.appender.Logfile.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.Logfile.layout.ConversionPattern=%d [%t] %-5p %c - %m%n
```

```
#Specifies to log any messages at the INFO level or higher emitted by  
loggers in the org.blueprint.seqhound sub-tree
```

```
log4j.logger.org.blueprint.seqhound=INFO
```

To learn more about log4j and its configuration, see:

<http://logging.apache.org/log4j/docs/documentation.html>

Example *shconfig.properties* files can be found in the root directories of source and binary tarball distributions.

## 2. Specifying data source types

The new seqhound java API framework is capable of accessing a Seqhound database server directly to fulfill requests, instead of having to go through a web service intermediate. Currently this is only available for the Text-Mining and Genbank ID conversion methods, which are new in release 3.2, but porting of older methods to a direct DB backend is planned for future releases. One configures whether or not to access a database directly in the Seqhound configuration properties, through the property named “accessDBDirectly”. This property is expected to be set to either true, if one wishes to access a database directly, or false if they only want to use the web service. For example, to specify that direct db access should be used, one would have the following line in *shconfig.properties*:

```
accessDBDirectly=true
```

Note that any queries which don't have a direct DB implementation will still work with the above setting, as they will use the web service implementation instead. As such, one must configure web service access even if they have specified DB access, otherwise they will be unable to use methods without direct DB implementations.

## 3. Method-specific data source overrides

While the *accessDBDirectly* property specifies which data sources types to be used in implementations for the API as a whole, one can override the data source used by individual methods in a method specific manner. To do this, simply set the method name

as a property in `shconfig.properties` to either 'remote' or 'db', depending on whether you want it to use remote web access or local db access. For example, the line:

```
SHoundGBAccFromGi=db
```

specifies that invocations of the API method `SHoundGBAccFromGi` should utilize direct local seqhound database access to retrieve the result. The line:

```
ShoundGiFromGBAcc=remote
```

specifies that invocations of the API method `ShoundGBAccFromGi` should query a remote seqhound web service server (either `jseqrem` or `seqrem`) to retrieve the result. These method-specific settings override what is specified by the *accessDBDirectly* property.

#### 4. Configuring Data Sources

The *accessDBDirectly* property specifies which data sources to utilize, but additional configuration is required to tell the application how to access those data sources, such as the URLs of remote `seqrem` servers and local database servers.

Remote `jseqrem` servers (the java servlet implementation of the seqhound web service) are specified with the *seqremURL* property. This should be set to the URL of the `jseqrem` web application's `jseqrem` servlet. To set this value to the `jseqrem` on the Seqhound project's public servers, enter the following line in your configuration file:

```
seqremURL=http://seqhound.blueprint.org/seqhound/jseqrem:8080
```

Similarly, remote classic `seqrem` servers (the classic C CGI based implementation of the seqhound web service) are specified with the *cseqremURL* property:

```
seqremURL=http://seqhound.blueprint.org/cgi-bin/seqrem
```

Configuration of direct database access (when *accessDBDirectly* is set to true) is a bit more involved, requiring the setting of several properties. First and foremost is the `dburl` property, which specifies the database server URL which will be used for connecting to the database. An example for this property would be:

```
dburl=jdbc:mysql://myserver:3306/seqhound?user=myusername&  
password=mypassword&autoReconnect=true
```

Additionally, one must specify the fully qualified class name of the jdbc driver class to use. eg:

```
dbDriverName=com.mysql.jdbc.Driver
```

Note that seqhound direct db access has only been tested with MySQL 4.0+ and 4.1+ database backends; please notify the Seqhound project if you have trouble using it with other database backends, as we wish to keep the code RDBMS neutral.

One must also specify what type of database connection management mode should be used. Seqhound java's db access layer can utilize a single jdbc database connection, or it can create and manage a database connection pool, which can greatly improve

performance for multi-threaded applications (such as web servlets). The choice of which mode to operate in is specified by the `dbMode` property. To specify use of a single jdbc connection, set:

```
dbMode=jdbc
```

To specify use of database connection pooling, set:

```
dbMode=dbcp
```

Three optional properties are used to configure the database connection pool. *dbMinIdle* specifies the minimum number of idle connections which the database connection pool(dbcp) should maintain. *dbMaxActive* specifies the maximum number of connections which the dbcp is allowed to create. *dbMaxIdle* specifies the maximum number of idle connections which the dbcp will allow before it starts closing idle connections. An example configuration for these properties would be:

```
dbMinIdle=4
```

```
dbMaxIdle=4
```

```
dbMaxActive=25
```

If not explicitly set, `dbMinIdle`, `dbMaxIdle` and `dbMaxActive` default to 4, 4 and 50, respectively.

It is recommended that you copy the tarball's example configuration file 'shconfig.properties' to your applications current working directory and modify it to suit your needs. (see recent changes below)

## Bugs

If you think you may have found a bug then please email [seqhound@blueprint.org](mailto:seqhound@blueprint.org) with details.



## Using the remote API for PERL.

These instructions were last updated on August 30, 2004

These instructions take you through the process of setting up a development environment that uses the SeqHound remote API for the PERL programming language.

There are three major steps that are outlined in detail below.

1. Install PERL and install the LWP::simple module
2. For Windows install NMAKE
3. Download or check out the most recent SeqHound PERL module and install it
4. Set up your project

An example script is provided below.

If you have problems at any step, please contact *seqhound@blueprint.org*.

1. Install Perl and the LWP::simple module

- a) Install Perl

You must have installed Perl. See <http://www.cpan.org/> for freely available software and installation instructions. For Windows see <http://www.activestate.com/Products/ActivePerl/>

- b) Install *LWP::simple*.

A default Perl installation has a module defined to download and install modules from CPAN which you can invoke for this purpose. Enter:

```
perl -MCPAN -e 'install LWP::Simple'
```

The script will indicate whether LWP::Simple is already installed and up to date.

2. For Windows platforms you will need a copy of NMake available from Microsoft  
<http://download.microsoft.com/download/vc15/Patch/1.52/W95/EN-US/Nmake15.exe>  
Run the downloaded exe to extract it.  
Copy both the *NMAKE.EXE* and the *NMAKE.ERR* file to your Perl bin directory, normally *C:\Perl\bin*.

3. Download or check out the SeqHound Perl module

These instructions tell you how to download and install the Perl module from the SourceForge ftp site. The SeqHound PERL module is available from:

[http://sourceforge.net/project/showfiles.php?group\\_id=17918&package\\_id=39608](http://sourceforge.net/project/showfiles.php?group_id=17918&package_id=39608)

Find where your PERL modules are kept. For example *perl5.6/lib*

If the directory does not exist, or if it does not contain *LWP.pm*, you can search where your Perl modules are installed.

---

(Note - the SeqHound module does not **have** to be there, but its just a good idea to put files where you, your system and others would expect them to be.)

---

- a) Download the *seqhound.perl.X.X.tar.gz* file where *X.X* is the version number.

For Windows, using a web browser go to using a web browser go to one of the locations listed under item 2.

Save the file to your Perl module directory and uncompress it with PKZIP or WinZip. You can delete *seqhound.perl.X.X.tar.gz* after uncompressing it.

In a Unix environment you could use:

```
ftp ftp.sourceforge.net
login as anonymous
cd pub/sourceforge/slrtools
get seqhound.perl.X.X.tar.gz
bye
gunzip seqhound.perl.X.X.tar.gz
tar -xvf seqhound.perl.X.X.tar
rm seqhound.perl.X.X.tar
```

- b) Change the name of the uncompressed directory from *perl* to *seqhound*  
c) Move to the *seqhound* directory and make the PERL module

To install into the default location, you will need root/administrator access.

For UNIX

```
perl Makefile.PL
make
make test
make install
```

To install into non-default location (optional)

```
perl Makefile.PL LIB=/home/your/local/perl/lib
PREFIX=/home/your/local/perl
make
make install
```

You can skip the **make test** step since this will likely fail.

For Windows

```
perl Makefile.pl
nmake
nmake test
nmake install
```

#### 4. Set up a test project

##### a) Create the following PERL script using a text editor and call it *sh-mytest.pl*

```
#!/usr/bin/perl -w

# you may have to change the path above, eg to
# /usr/perl or /bin/perl, based on your system.

use strict;
use SeqHound;

# Initialize the Seqhound system.
# FALSE means that obsolete/outdated sequences from NCBI
# will not be queried
# Change to TRUE if obsolete/outdated sequences required
SHoundInit("FALSE", " sh-mytest-perl") or die "SHoundInit failed.\n";

print "****Starting Program\n";

my $id = "CAA28783";
print "Test SHoundFindAcc\n";
print "ID $id = Acc ", SHoundFindAcc($id), " \n";

# Close the SeqHound system
my $aa = SHoundFini();
print "****SeqHound closed: $aa\n";
```

##### b) Run the PERL script

```
perl sh-mytest.pl
```

If everything is working correctly, you should see

```
****Starting Program
Test SHoundFindAcc
ID CAA28783 = Acc 56756
****SeqHound closed: TRUE
```

### Troubleshooting

#### 1. Error messages

Error messages are written to the log file, *shoundlog*, found in the same directory as your script.

#### 2. Test script

There is a script called *test.pl* which will test all the API functions. It takes a test file called *input*. Test results will be written to *perl\_test.log* and *test\_summary.log*.

To run the test script:

```
perl test.pl
```

### Documentation

API documentation is included on the SeqHound website

[http://www.blueprint.org/seqhound/api\\_help/apifunctslist.html](http://www.blueprint.org/seqhound/api_help/apifunctslist.html) and is available in POD format in the *README\_API.pod* included with the package.

**Bugs**

If you think you may have found a bug then please email [seqhound@blueprint.org](mailto:seqhound@blueprint.org) with details.

## Using the remote API for Bioperl

These instructions were last updated on August 30, 2004

This is the first official release of the SeqHound Bioperl module. Note that this package is distributed and maintained by the Blueprint Initiative. It is not an official part of the Bioperl release. In future, functionality contained in this package could be included as part of Bioperl. Contact [seqhound@blueprint.org](mailto:seqhound@blueprint.org) for more details.

### Prerequisites

The SeqHound bioperl module requires the following additional modules and libraries:

1. Perl 5.006
2. LWP::Simple
3. The Bioperl collection

### Installation

1. Perl 5.006 & LWP::Simple can be downloaded at <http://www.cpan.org/>

For installation instructions please consult the documentation available on the cpan website.

2. The Bioperl collection can be downloaded at <http://www.bioperl.org/>

For example on UNIX

```
wget http://bioperl.org/DIST/current_core_stable.tar.gz
tar zxvf current_core_stable.tar.gz
```

Bioperl is dependent on several non-Perl applications. These may be installed using the Perl command:

```
perl -MCPAN -e "install Bundle::BioPerl"
```

You will be prompted on customizing the install packages. You can use the defaults by hitting **RETURN** at each prompt.

- a) To install Bioperl in the default location you must have root access.

```
cd bioperl-X.X
perl Makefile.PL
make
make test
make install
```

- b) To install Bioperl in a non-default location (optional)

This still requires the Bioperl dependencies (which requires you to have root/administrator access). Once the dependencies are installed, you can install the Bioperl packages anywhere.

```
cd bioperl-X.X
perl Makefile.PL LIB=/home/your/local/bioperl/lib
PREFIX=/home/your/local/bioperl
```

```
make
```

```
make install
```

You can skip the **make test** step since this will likely fail.

For more detailed instructions regarding Bioperl please see the documentation on the Bioperl website.

3. SeqHound Bioperl module is available from SourceForge:

[http://sourceforge.net/project/showfiles.php?group\\_id=17918&package\\_id=39608](http://sourceforge.net/project/showfiles.php?group_id=17918&package_id=39608)

and the SeqHound Blueprint ftp site: <ftp://ftp.blueprint.org/pub/SeqHound/>

For example on UNIX

```
wget ftp://ftp.blueprint.org/pub/SeqHound/seqhound-  
bioperl-x.xx.tar.gz
```

```
tar zxvf seqhound-bioperl-x.xx.tar.gz
```

- a) To install in the default location you must have root/administrator access.

```
cd seqhound-bioperl-x.xx
```

```
perl Makefile.PL
```

```
make
```

```
make test
```

```
make install
```

- b) To install in a non-default location (optional)

```
cd seqhound-bioperl-x.xx
```

```
perl Makefile.PL LIB=/home/your/local/bioperl/lib  
PREFIX=/home/your/local/bioperl
```

```
make
```

```
make install
```

You can skip the **make test** step since this will likely fail.

---

NOTE: If you install Bioperl and SeqHound in a non-default location, you must add the following statement to your Perl scripts:

```
#!/usr/bin/perl  
use lib "/home/your/local/bioperl/lib";  
use Bio::SeqHound::SeqHound;  
#your code here
```

Alternatively, in configuration file

```
setenv PERL5LIB /home/your/local/bioperl/lib  
and add the following statement to your scripts:  
#!/usr/bin/perl
```

---

By default all the public functions in SeqHound are exported.

## Documentation

POD documentation is available in the source code.

**Bugs**

If you think you may have found a bug then please email [seqhound@blueprint.org](mailto:seqhound@blueprint.org) with details.

## 4. Setting up SeqHound locally.

### 4.1 Overview.

This section describes how one can set up the SeqHound system on your own hardware using freely available SeqHound executables. These executables will allow you to build and update the SeqHound database as well as run a web-interface and a remote API server.

Section 4.2 should be reviewed first for system requirements before attempting to install the SeqHound system.

Section 4.3 tells you how to download executables from the SeqHound ftp site for your platform and operating system. SeqHound code may also be downloaded from this site.

Section 4.4 describes how SeqHound code may be compiled on your own hardware using the freely available code available on the SeqHound ftp site. This step is only required if SeqHound executables are not available for your platform or if you want to make use of the local programming API. If you obtain SeqHound executables from the ftp site and want to build your local SeqHound database, you still need to go through Steps 8, 9, 10, 11 and 13 in this section which are how to install MySQL server and ODBC driver.

Section 4.5 contains detailed instructions for using the executables to build the SeqHound data tables and for setting up the SeqHound web-interface and remote API server.

Section 4.6 contains instructions for setting up SeqHound for ODBC on Solaris

Section 4.7 contains instructions for setting up SeqHound on Linux (SeqHound in a Box)



## 4.2 SeqHound system requirements.

Before attempting to set up SeqHound yourself, you should review the system requirements listed below. The SeqHound system is able to run on a number of operating systems (we recommend and can best support a UNIX operating system like Sun Solaris or Red Hat Linux). Setting up SeqHound will require approximately 600 GB of disk space (see below).

Questions about system requirements, compilation, setup and maintenance can be addressed to [seqhound@blueprint.org](mailto:seqhound@blueprint.org). We will do our best to address all queries but resources may not allow us to solve all problems arising on all possible set ups.

### OS and hardware architecture

SeqHound code is compiled on the following platforms on a daily basis. Blueprint production SeqHound is compiled and run on Sun Ultra - Sun Solaris (version 8). We have also compiled and run SeqHound on Red Hat Linux. The other platforms have not been tested.

Release versions of SeqHound executables are available for.

x86 architecture	Red Hat Linux
Sun Ultra	Sun Solaris (version 8)
PowerPC architecture	MacOS X

Please contact us at [seqhound@blueprint.org](mailto:seqhound@blueprint.org) if you would like the following executables.

x86 architecture	FreeBSD
x86 architecture	QNX
x86 architecture	Windows NT
PowerPC architecture	PPC Linux
SGI	Irix 6
Alpha architecture	Compaq Alpha OS
HPPA 2.0 architecture	HPUX 11.0
HPPA 1.1 architecture	PA-RISC Linux

### Memory (RAM)

We recommend a minimum of **1 GB** of RAM to run the SeqHound executables.

**Hard Disk****Source code and executables**

<b>Component</b>	<b>Image Size</b>
SeqHound Source and compiled	220.0 MB
NCBI Toolkit	271.0 MB
NCBI C++ Toolkit	490.3 MB
bzip2 Library	1.4 MB
slri lib	1.2 MB
slri lib_cxx	9.4 MB
<b>Source code and executables (total)</b>	1000.0 MB approx

**Database**

<b>Component</b>	<b>Image Size</b>
data tables	250 GB
data tables backup	250 GB
<b>Database (total)</b>	600 GB*

\*600GB includes 250 GB for a single copy of the SeqHound data tables. The SeqHound system includes a second copy of the data tables used for back up and updating. We suggest a minimum of **600 GB** for SeqHound installation. This allows for yearly growth of the data tables as well as for a RAID5 disk configuration.

**Other Software**

Apache Webserver	See <a href="http://www.apache.org/">http://www.apache.org/</a> for software installation for you platform.
Perl	See <a href="http://www.cpan.org/">http://www.cpan.org/</a> for installation for your platform. Required modules include <i>Net/FTP.pm</i> , <i>sun4-solaris-64/DBI.pm</i>

**Compiling SeqHound Code yourself.**

It is not necessary to compile SeqHound executables yourself; the system may be set up using the executables provided on the ftp site. However, if you wish to make use of the local API then you must compile SeqHound yourself.

**Database engine**

ODBC compliant database engines	Blueprint uses MySQL as an ODBC compliant database engine.
---------------------------------	--

**Library dependencies**

NCBI Toolkit	from <a href="ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools/">ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools/</a>
NCBI C++ Toolkit	from <a href="ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools/">ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools/</a>
bzip2 Library	from <a href="http://sourceforge.net/projects/slrtools/">http://sourceforge.net/projects/slrtools/</a>
slri lib	from <a href="http://sourceforge.net/projects/slrtools/">http://sourceforge.net/projects/slrtools/</a>
slri lib_cxx	from <a href="http://sourceforge.net/projects/slrtools/">http://sourceforge.net/projects/slrtools/</a>

### 4.3 Obtaining SeqHound executables and/or code.

You will require about 300 MB of disk space to store the SeqHound code and compiled executables.

1. Make a new directory where you will store the new SeqHound executables and or code.

```
mkdir compile
```

Move into this directory and set an environment variable called `COMPILE` to point to this directory.

```
cd compile
```

```
setenv COMPILE `pwd`
```

(where ```) is a single back-quote)

2. Go to the SeqHound ftp site

From the *compile* directory, type:

```
ftp ftp.blueprint.org
```

When prompted for a name enter **anonymous**

When prompted for a password type your email address: **myemail@home.com**

```
cd pub/SeqHound/
```

3. Get the SeqHound code.

```
cd Code
```

```
bin
```

```
get seqhound_X.X.tar.gz
```

where **X.X** is the date the most recent code release number.

4. Get SeqHound precompiled binaries for your platform.

---

(If you will be compiling the SeqHound executables yourself (see [section 4.4](#) below) you can skip this step.

---

```
cd ../precompiled
```

```
dir
```

Each directory contains precompiled binaries for the indicated operating system.

Move to the directory corresponding to your operating system. For example:

```
cd Execs
```

```
bin
```

```
get platform.tar.gz
```

Close the ftp session by typing:

```
bye
```

5. Unpack the SeqHound code.

```
uncompress seqhound_X.X.tar.gz  
tar xvf seqhound_X.X.tar  
ls -la
```

You should see several directories here

```
bzip2  
ncbi  
slri
```

6. Set the SLRI environment variable

Move to the *slri* directory and set the environment variable “SLRI” to point to this directory.

```
cd $COMPILE/slri  
setenv SLRI `pwd`
```

7. Unpack the executables.

You can skip this step if you skipped step 4.

```
uncompress seqhound_execs.tar.Z  
tar xvf seqhound_execs.tar  
ls -la
```

You should see another directory here called

```
execs
```

Move everything in the *execs* directory into the *slri/seqhound/build/odbc* directory.

```
cd $COMPILE/execs  
mv * $COMPILE/slri/seqhound/build/odbc .
```

If you downloaded and unpacked the executables, you can skip section 4.4 and continue with section 4.5.

---

Note: At the time of writing, it is also possible to download the SeqHound code as a compressed file from the Sourceforge site at <http://sourceforge.net/projects/slrtools/>. (for example: *seqhound-X.X.tar.gz* where *X.X* is the release number).

---

## 4.4 Compiling SeqHound executables on Solaris

These instructions describe how to compile SeqHound running on the Solaris platform. They may be used as a guide for compiling SeqHound code on other platforms. Instructions are similar for Linux and differences are noted.

### Using these instructions

These instructions assume that:

- You have downloaded the SeqHound code from the ftp server. See [section 4.3](#).
- You have set environment variables called `COMPILE` and `SLRI`. See steps 4.3.1. and 4.3.6.
- You are using the `c` shell.
- On Linux platforms, to compile SeqHound libs with ODBC support you also need `unixODBC-devel` package which contains the `sql.h` + other libs/headers required to compile SeqHound libs with ODBC support. This is not needed to run SeqHound, just to compile it.

These instructions were tested on a Sun-Ultra architecture running a Sun Solaris OS (version 8). The system information for the test-box (results of a “`uname -a`” call) were:

```
SunOS machine_name 5.8 Generic_108528-01 sun4u sparc
SUNW,Ultra-4
```

#### 1. Download the NCBI toolkit

SeqHound is dependent on code in the NCBI toolkit

Move to the *compile* directory and ftp to the NCBI ftp site:

```
cd $COMPILE
ftp ftp.ncbi.nlm.nih.gov
```

When prompted for a name enter **anonymous**

When prompted for a password type **myemail@home.com**

```
cd toolbox/ncbi_tools
```

Make a note of the *FAQ.html* and the *readme.htm* files.

Change your transfer type to binary and get the zipped directory called *ncbi.tar.Z*.

```
bin
```

```
get ncbi.tar.Z
```

Close the ftp session by typing:

```
bye
```

Uncompress the toolkit.

```
uncompress ncbi.tar.Z
```

```
tar xvf ncbi.tar
```

2. Edit the platform make file.

Go to the platform directory and locate the file with a “.mk” extension that applies to your platform. In Solaris the file is “*solaris-gcc.ncbi.mk*” and in Linux the file is *linux-x86.ncbi.mk*.

```
cd $COMPILE/ncbi
```

```
cd platform
```

In Linux *linux-x86.ncbi.mk* replace the line `/home.coremake/ncbi` with `${NCBI}`

Use the following line (a Perl command) to replace the string in the Solaris file `/netopt/ncbi_tools/ver0.0/ncbi` with the string `${NCBI}` in the *solaris-gcc.ncbi.mk* file:

```
perl -p -i.bak -e  
's|/netopt/ncbi_tools/ver0.0/ncbi|\\${NCBI}|g' solaris-  
gcc.ncbi.mk
```

so for instance, the line

```
NCBI_INCDIR = /netopt/ncbi_tools/ver0.0/ncbi/include
```

Will become:

```
NCBI_INCDIR = ${NCBI}/include
```

You could also edit this file in hand using a text editor if you don’t have Perl installed.

Copy the file up one level to the *ncbi* directory and rename it “*ncbi.mk*”

```
cp solaris-gcc.ncbi.mk ../ncbi.mk
```

3. Set environment variables in preparation for the toolkit build.

Move back to the *ncbi* directory and set the environment variable `NCBI` to point to that directory

```
cd $COMPILE/ncbi
```

```
setenv NCBI `pwd`
```

or

```
setenv NCBI "$COMPILE/ncbi"
```

check this by typing

```
echo $NCBI
```

the value shown will replace `${NCBI}` in the “*solaris-gcc.ncbi.mk*” file that you modified in the above step when the make file is run.

You must also make certain that environment variables have been set that indicate that the gcc compiler is to be used

```
setenv CC gcc
```

Finally, paths to the gcc compiler and the archive executable *ar* should be added to your `PATH` variable:

```
setenv PATH /usr/local/bin:/usr/ccs/bin:${PATH}
```

You can check all of your environment variables by typing

```
printenv | sort
```

At this point, the relevant environment variables should be something like this:

```
> printenv | sort
```

```
CC=gcc
```

```
COMPILE=/export/home/your_user_name/compile
```

```
NCBI=/export/home/your_user_name/compile/ncbi
```

```
OSTYPE=solaris
```

```
PATH=/usr/local/bin:/usr/ccs/bin:/usr/bin:/usr/ucb:/etc:.
```

You can permanently set these environment variables by editing your appropriate configuration file, for example, *.tcshrc* file (not covered here).

If you want, you can read the readme file in the *make* directory.

```
cd make
```

```
more readme.unx
```

#### 4. Build the NCBI toolkit

Move back up to the *compile* directory and run the make command.

```
cd $COMPILE
```

```
./ncbi/make/makedis.csh |& tee out.makedis.txt
```

---

Note: to build Solaris 64 bit binaries add the following to the command line:

```
SOLAIRS_MODE=64 ncbi/make/makedis.csh
```

---

This runs a c-shell script to make the toolkit and tees the output to the screen and a log file “*out.makedis.txt*”. It is safe to ignore the multiple error messages that you may see.

At the end of a successful build you will see

```
*****
*The new binaries are located in ./ncbi/build/ directory*
*****
```

The *ncbi.tar* file can be removed from the “*compile*” directory after the successful build process has been completed.

---

Note: For Solaris UNIX only, SeqHound API SHoundGetGenBankff and SHoundGetGenBankffList breaks due to a bug in the NCBI library file *ncbistr.c* (in directory *ncbi/corelib* and *ncbi/build*). To fix the problem, replace all the code inside the function `Nlm_TrimSpacesAroundString()` in the file *ncbistr.c* with the following text

```
char *ptr, *dst, *revPtr;
int spaceCounter = 0;
ptr = dst = revPtr = str;

if ( !str || str[0] == '\0' )
    return str;

while ( *revPtr != '\0' )
    if ( *revPtr++ <= ' ' )
        spaceCounter++;

if ( (revPtr - str) <= spaceCounter )
{
    *str = '\0';
    return str;
}

while ( revPtr > str && *revPtr <= ' ' ) revPtr--;
while ( ptr < revPtr && *ptr <= ' ' ) ptr++;
while ( ptr <= revPtr ) *dst++ = *ptr++;
*dst = '\0';

return str;
```

---

## 5. Make the bzip2 library

The bzip2 code was downloaded as part of the seqhound code in step 4.3.3 above. Move to the *bzip2* directory and run the make file.

```
cd $COMPILE/bzip2
make -f make.bzlib
```

## 6. Set the BZDIR environment variable.

```
cd $COMPILE/bzip2
setenv BZDIR `pwd`
```

## 7. In your home directory, add the following environment parameters to the appropriate configuration file such as *.bashrc* or *.bash\_profile*. Text in italics should be changed to the correct path on your machine that points to directory having DBI.pm:

```
export NCBI=$COMPILE/ncbi
export BZDIR=$COMPILE/bzip2
export SLRI=$COMPILE/slri
export CC="gcc -ansi"
export VIBLIBS="-L/usr/X11R6/lib -lXm -lXpm -lXmu -lXp -
lXt -lX11 -lXext"
```



```
export
PERL5LIB=/usr/local/lib/perl5/site_perl/5.8.3/sun4-
solaris-64
```

8. Install MySQL server and create database "seqhound".

SeqHound is built and tested in MySQL version 4.0.20. You can download MySQL from <http://dev.mysql.com/downloads/mysql/4.0.html> and follow the manual at <http://dev.mysql.com/doc/mysql/en/index.html> to install MySQL on your server. The data directory where the MySQL server points to should have 600 GB for a full SeqHound database. After MySQL is installed, you need to log into MySQL and create database "seqhound":

```
create database seqhound;
```

---

Note that ";" must be used at the end of all MySQL statements.

---

9. Install ODBC driver:

---

Note that for Linux platforms, unixODBC package needs to be installed prior to the ODBC driver otherwise the following error will occur:

```
error: Failed dependencies:
      libodbcinst.so.1 is needed by MyODBC-3.51.09-1
```

- 
- a) Go to web site: [http://dev.mysql.com/doc/connector/odbc/en/faq\\_2.html](http://dev.mysql.com/doc/connector/odbc/en/faq_2.html)
  - b) Find and download RPM distribution of ODBC driver *MyODBC-3.51.07-1.i586.rpm*.
  - c) As user "root", install the driver.  
For first time installation  

```
rpm -ivh MyODBC-3.51.01.i386-1.rpm
```

  
For upgrade  

```
rpm -Uvh MyODBC-3.51.01.i386-1.rpm
```
  - d) The library file *libmyodbc3*. will be installed in directory */usr/lib* or */usr/local/lib*.

10. Set up configuration file for ODBC driver.

Create a configuration file called *.odbc.ini* in your home directory with the following content:

Edit the file called *.intrezrc* in directory *slri/seqhound/config/*.

```
[mysqlsh]          ← header must not be used for other sections
Description  = MySQL ODBC 3.51 Driver DSN
Trace        = Off
TraceFile    = stderr
Driver       = /usr/lib/libmyodbc3.so
DSN          = mysqlsh ← same as the header name
SERVER       = my_server
PORT         = my_port
USER         = my_id
PASSWORD     = my_pwd
DATABASE     = seqhound ← database name
```

Text in ***italics*** should be changed. Text */usr* in the value of variable *Driver* should be changed to the path where *unixodbc* resides. Text ***my\_server*** should be changed to the IP address or the server name of the MySQL server. Text ***my\_port*** should be changed to port number of the MySQL instance. Text ***my\_id*** and ***my\_pwd*** should be replaced by your user id and password to the MySQL database.

---

Note that the values for the headers such as *DSN*, *USER*, *PASSWORD* and *DATABASE* must be less than 9 characters.

---

## 11. Set up ODBC related variables:

```
setenv ODBC path_to_unixodbc
```

Where ***path\_to\_unixodbc*** should be replaced by the path of the UnixODBC driver on your machine.

In your home directory, add parameter “LD\_LIBRARY\_PATH” to the appropriate configuration file such as *.bashrc* or *.bash\_profile*:

```
export LD_LIBRARY_PATH =  
/usr/local/unixodbc/lib:/usr/local/unixodbc/odbc/lib:/usr  
/local/mysql/lib/mysql:/usr/local/mysql/lib/mysql/lib
```

The value of variable “LD\_LIBRARY\_PATH” should have all the paths that have the library files *libodbc\**, *libmyodbc\**, and *libmysqlclient\**

## 12. Build the SeqHound executables

Move to the compile directory and list all the files in the directory:

```
cd $COMPILE  
ls
```

You should see:

```
> ls  
bzip2  
ncbi
```

```
slri
out.makedis.txt
```

Before proceeding you should check your environment variables

```
printenv | sort
```

to ensure that correct paths have been specified for each of the following variables:

```
NCBI
SLRI
ODBC
BZDIR
```

---

Note: at the time of writing, you may have to alter the contents of *slri/seqhound/seqhound.mk* and *slri/lib/slri.mk* to ensure that the following lines occur:

```
BZDIR = BZIP2
BZINC = BZDIR
```

---

Compile the SLRI libraries using the following commands:

```
cd $SLRI/lib
make -f make.slrilib
make -f make.slrilib odbc
```

The above commands will build the SLRI libraries needed by SeqHound.

The make files which you are about to invoke call on these variables therefore the paths must be correct. Move to the make directory for SeqHound and run the *makeall* script. Capture the output.

```
cd $SLRI/seqhound
./makeallsh
```

A description of how to use the *makefile* will be shown. The first parameter indicates what database backend is to be used for the build (only *odbc*). The second parameter indicates what SeqHound programs are to be made (a choice of *all*, *cgi*, *domains*, *examples*, *genomes*, *go*, *locuslink*, *parsers*, *scripts*, *taxon*, *updates*).

```
./makeallsh odbc all > out.makeseqhound.txt
```

It is safe to ignore the multiple warning messages that you may see.

After this has finished running, move to the *build* directory in the *seqhound* directory where you will find the executables for SeqHound.

```
cd build/odbc
ls -l
```

You will see

```
>ls -l
addgoid
cbmmdb
chrom
clustmask
clustmasklist
comgen
fastadom
gen2fasta
gen2struc
goparser
goquery
importtaxdb
isshoundon
llgoa
llparser
llquery
mother
pdbrep
precompute
redund
seqrem
shrest
update
vastblst
wwwseekgi
```

13. Set up the SQL files that create tables.

```
cd $SLRI/seqhound/sql
```

In each of files *core.sql*, *redund.sql*, *ll.sql*, *taxdb.sql*, *gendb.sql*, *strucdb.sql*, *cddb.sql*, *godb.sql*, *rps.sql*, *nbr.sql*, there is a line close to the beginning of each file:

```
#use testsql;
```

This line should be changed to

```
use seqhound;
```

## 4.5 Building the SeqHound system on Solaris

Using these instructions

These instructions show how the SeqHound executables may be used to build the SeqHound system under a Solaris 8 OS. These instructions may also be used as a guide for setting up SeqHound under other operating systems. These instructions assume that:

- You have downloaded the latest release version of the SeqHound code (see step 4.3.3)
- You have successfully installed MySQL
- You have successfully compiled the SeqHound code yourself (*section 4.4*)

OR

you have downloaded the SeqHound executables for your platform and operating system (section 4.3.4).

- You have set environment variables called `COMPILE` and `SLRI` (see steps 4.3.1 and 4.3.6).
- You have a default install of an Apache server running. See <http://www.apache.org/> for freely available software and instructions for your platform.
- You have installed Perl. See <http://www.cpan.org/> for freely available software and installation instructions.
- You have at least 300 MB space available in a directory where you can check out code and compile it.
- You have at least **600 GB** available for the SeqHound executables and data tables. See *section 4.2*.

These instructions were tested on a Sun Ultra machine running the Sun-Solaris 8 OS. The system information for the test-box (results of a “`uname -a`” call) were:

```
SunOS machine_name 5.8 Generic_108528-01 sun4u sparc  
SUNW,Ultra-4
```

These instructions assume that you are using the `c` shell. Syntax may differ for some commands in other shells.

---

Note: These instructions begin with ‘step 14’.

---

#### 14. Prepare to build the SeqHound database.

Create a new directory where you will set up SeqHound.

```
mkdir seqhound
```

Set the environment variable SEQH to point to this directory.

```
cd seqhound
```

```
setenv SEQH `pwd`
```

Move to this directory and create new directories

```
cd seqhound
```

```
mkdir 1.core.files
```

```
mkdir 2.redund.files
```

```
mkdir 3.taxdb.files
```

```
mkdir 4.godb.files
```

```
mkdir 5.lldb.files
```

```
mkdir 6.comgenome.files
```

```
mkdir 7.mmdb.files
```

```
mkdir 8.hist.files
```

```
mkdir 9.neighbours.files
```

```
mkdir 10.rpsdb.files
```

```
mkdir precompute
```

The numbered directories will hold parsers and files required for the build of the SeqHound data tables. Directory “*precompute*” will hold the precomputed data of the database.

Move to each of the numbered directories and copy all of the scripts and executables required for the build.

```
cd $SEQH/1.core.files
cp $SLRI/seqhound/sql/core.sql .
cp $SLRI/seqhound/scripts/asnftp.pl .
cp $SLRI/seqhound/scripts/seqhound_build.sh .
cp $SLRI/seqhound/build/odbc/mother .
cp $SLRI/seqhound/build/odbc/update .
cp $SLRI/seqhound/config/.intrezrc .

cd $SEQH/2.redund.files
cp $SLRI/seqhound/sql/redund.sql .
cp $SLRI/seqhound/scripts/nrftp.pl .
cp $SLRI/seqhound/build/odbc/redund .

cd $SEQH/3.taxdb.files
cp $SLRI/seqhound/sql/taxdb.sql .
cp $SLRI/seqhound/scripts/taxftp.pl .
cp $SLRI/seqhound/build/odbc/importtaxdb .

cd $SEQH/4.godb.files
cp $SLRI/seqhound/sql/godb.sql .
cp $SLRI/seqhound/scripts/goftp.pl .
cp $SLRI/seqhound/build/odbc/goparser .

cd $SEQH/5.lldb.files
cp $SLRI/seqhound/sql/ll.sql .
cp $SLRI/seqhound/scripts/llftp.pl .
cp $SLRI/seqhound/build/odbc/llparser .
cp $SLRI/seqhound/build/odbc/addgoid .

cd $SEQH/6.comgenomes.files
cp $SLRI/seqhound/sql/gendb.sql .
cp $SLRI/seqhound/scripts/genftp.pl .
cp $SLRI/seqhound/scripts/humoasn.pl .
cp $SLRI/seqhound/scripts/humouse_build.sh .
cp $SLRI/seqhound/scripts/comgencron_odbc.pl .
cp $SLRI/seqhound/scripts/shconfig.pm .
cp $SLRI/seqhound/genomes/gen_cxx .
cp $SLRI/seqhound/genomes/pregen.pl .
```

```
cp $SLRI/seqhound/genomes/gen.pl .
cp $SLRI/seqhound/genomes/ncbi.bacteria.pl .
cp $SLRI/seqhound/build/odbc/chrom .
cp $SLRI/seqhound/build/odbc/comgen .
cp $SLRI/seqhound/build/odbc/mother .
```

```
cd $SEQH/7.mmdb.files
cp $SLRI/seqhound/sql/strucdb.sql .
cp $SLRI/seqhound/scripts/mmdbftp.pl .
cp $SLRI/seqhound/config/.mmdbrc .
cp $SLRI/seqhound/config/.ncbirc .
cp $SLRI/seqhound/build/odbc/cbmmdb .
```

```
cd $SEQH/8.hist.files
cp $SLRI/seqhound/build/odbc/histparser .
```

Open the *.intrezrc* file with a text editor like pico and edit.

```
cd $SEQH/1.core.files
pico .intrezrc
```

An example *.intrezrc* file follows. Lines preceded by a semi-colon are comments that explain what the settings are used for and their possible values.

Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in the *.odbc.ini* file you set up in Step 10 in section 4.4. For variable path and indexfile in section [precompute], replace the text in ***italics*** with the absolute path of directory “precompute” you just created.

---

---

Warning: This file may have wrapped lines. Take care when editing this file that you do not break any of the lines (i.e. introduce any unwanted carriage returns).

---

---



```

-----example .intrezrc begins-----
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in .odbc.ini_file
database=seqhound
local=
[config]
;the executable the cgi runs off of.
CGI=wwwseekgi
[precompute]
;precomputed taxonomy queries
MaxQueries = 100
MaxQueryTime = 10
QueryCount = 50
path = /seqhound/precompute/
indexfile = /seqhound/precompute/index
[sections]
;indicated what modules are available in SeqHound
;1 for available, 0 for not available
;gene ontology hierarchy
godb = 1
;locus link functional annotations
lldb = 1
;taxonomy hierarchy
taxdb = 1
;protein sequence neighbours
neigdb = 1
;structural databases
strucdb = 1
;complete genomes tracking
gendb = 1
;redundant protein sequences
redundb = 1
;open reading frame database
;currently not exported to outside users of SeqHound
cddb = 0
;RPS-BLAST domains
rpsdb = 1
;DBXref Database Cross_Reference
dbxref = 0

[crons]
;customizable variables in cron jobs
;NOTE: all paths must end in '/'
pathupdates=./
pathinputfiles=./
pathinputfilescomgen=./
mail=user\@host.org
defaultrelease=141
pathflags=./
-----example .intrezrc ends-----

```

This file should be copied to other directories used during the build process:

```

cp .intrezrc $SEQH/2.redund.files/.
cp .intrezrc $SEQH/3.taxdb.files/.
cp .intrezrc $SEQH/4.godb.files/.
cp .intrezrc $SEQH/5.lldb.files/.
cp .intrezrc $SEQH/6.comgenome.files/.
cp .intrezrc $SEQH/7.mmdb.files/.
cp .intrezrc $SEQH/8.hist.files/.
cp .intrezrc $SEQH/9.neighbours.files/.
cp .intrezrc $SEQH/10.rpsdb.files/.

```

#### 15. Build the core module of SeqHound.

Building the core module (basically all of the sequence data tables) is not optional. The rest of the modules are optional if there is a need to spare resources or administrative efforts but the corresponding API functionality will not be present.

```
cd $SEQH/1.core.files
```

Create the core tables in the database

Make sure file *core.sql* has line `use seqhound` close to the beginning of the file.

```
mysql -u my_id -p -P my_port -h my_server < core.sql
```

Where *my\_id*, *my\_port* and *my\_server* should be replaced by your userid for the database, the port of the database and the IP address or the server name of the database server respectively. You will be prompted to enter your password.

This creates core tables *acddb*, *asndb*, *nucprot*, *parti*, *pubseq*, *sendb*, *sengi*, *taxgi*, *bioentity*, *bioname*, *secondrefs*, *bioentitytype*, *nametype*, *rules*, *fieldtype* and *histdb*.

If you are building a full-instance of the SeqHound database then run the *asnftp.pl* script while in the *build* directory:

```
./asnftp.pl
```

---

Note that any command in these instructions can be run as a 'nohup' to prevent the process from ending if your connection to the machine should be lost. For example:

```
nohup ./asnftp.pl &
```

---

If you only want to build a small test version of the database then manually download a single file. For example:

```
ftp ftp.ncbi.nih.gov
```

When prompted for a name enter **anonymous**

When prompted for a password type *myemail@home.com*

```
cd refseq/cumulative
```

```
bin
```

```
get rscu.bna.Z (do not uncompress this file)
```

```
bye
```

The *asnftp.pl* script downloads all of the GenBank sequence records (in binary ASN.1 format) required to make an initial build of the SeqHound core module. This script will take approximately 24 hours to run and will consume 14 GB of disk space.

---

Note that all scripts are described in detail in section 5.

---

Two other files are generated by this script:

*asn.list* is a list of the sequence files that the script intends to download.

*asnftp.log* is where the script logs error messages during execution time.

If you open another session with the machine where you are building SeqHound, you can check how far along *asnftp.pl* is by comparing the number of lines in the *asn.list* file

```
grep ".aso.gz" asn.list | wc -l
```

to the number of lines in the build directory (number of files actually downloaded so far)

```
ls *.aso.gz | wc -l
```

Once *asnftp* has finished, these two numbers should be the same.

Run the seqhound build script. Before running this script, make certain that the *.intrezrc* file, in the same directory, and *.odbc.ini*, in your home directory, have correct configuration values. (see steps 10 in section 4.4 and step 14 in the current section). This parser MUST be given a single parameter that represents the release version of GenBank. You can find the release number in the file:

```
ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily/Last.Release.
```

```
./seqhound_build.sh 141
```

*seqhound\_build.sh* executes the mother parser over all source files and populates tables accddb, asndb, nucprot, parti, pubseq, sendb, sengi, taxgi, bioentity, bioname, secondrefs, bioentitytype, nametype, rules and fieldtype. This will take about 75 hours. Table histdb is still empty at this stage. It is populated in Step 25.

Parser mother creates a log file for every *\*.aso* file that it parses. These log files are located in a subdirectory called "logs" and are named "rsnc0506run" where "rsnc0506" is the name of the file that was being processed.

---

While *seqhound\_build.sh* is running, you can move on to steps 16-18.

---

Once *seqhound\_build.sh* has finished you can test that all of the files were properly processed by showing that the results of

```
cd logs
```

```
grep "Done" | wc -l
```

is the same as

```
ls *run | wc -l
```

is the same as

```
cd ..
```

```
ls *.aso.gz | wc -l
```

The *seqhound\_build.sh* script unzips *.aso.gz* files before feeding them as input to the mother program. *seqhound\_build.sh* then rezip the file after mother is done with it. If for some reason, the build should crash part way through, you have to

- a) recreate core tables using *core.sql* (see above) and
- b) search for any unzipped (*\*.aso* files) in the *build* directory and rezip them
- c) restart *seqhound\_build.sh*.

Once the *seqhound\_build.sh* script has finished, you should move all of the *\*.aso.gz* files into a directory where they will be out of the way:

```
mkdir asofiles  
mv *.aso.gz asofiles/.
```

#### 16. Build the redundb module.

```
cd $SEQH/2.redund.files
```

Create table redund in the database.

Make sure file *redund.sql* has the line `use seqhound` close to the beginning of the file.

```
mysql -u my_id -p -P my_port -h my_server < redund.sql
```

Where *my\_id*, *my\_port* and *my\_server* should be replaced by your userid for the database, the port of the database and the IP address or the server name of the database server respectively. You will be prompted to enter your password.

This creates table redund in the database.

Run the *nrftp.pl* script to download the FASTA nr database of proteins (<ftp://ftp.ncbi.nlm.nih.gov/blast/db>).

```
./nrftp.pl
```

*nrftp.pl* generates a log file *nrftp.log* that informs you what happened. If everything went ok, the last two lines should read:

```
Getting nr.gz  
closing connection
```

A new file should appear in the *build* directory called *nr.Z*. You will have to unpack this file by typing:

```
gunzip nr.gz
```

Run the redund parser to make the redund table of identical protein sequences.

Before running this script, make certain that the *.intrezrc* file in the same directory and *.odbc.ini* in your home directory have correct configuration values (see step 10 in section 4.4 and step 14 in the current section).

```
./redund -i nr -n F
```

redund generates the log file *redundlog*. If everything went ok, the only line in this file should be:

```
NOTE: [000.000] {redund.c, line 259} Done.
```

And about 3 millions records will be inserted into table redund.

#### 17. Build the taxdb module

Create tables of the taxdb module in the database.

```
cd $SEQH/3.taxdb.files
```

Make sure file *taxdb.sql* has line use seqhound close to the beginning of the file.

```
mysql -u my_id -p -P my_port -h my_server < taxdb.sql
```

Where *my\_id*, *my\_port* and *my\_server* should be replaced by your userid for the database, the port of the database and the IP address or the server name of the database server respectively. You will be prompted to enter your password.

This creates tables taxdb, gcodedb, divdb, del, merge in the database.

Run the *taxftp.pl* script to download taxonomy info from the NCBI (<ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz>).

```
taxftp.pl
```

*taxftp.pl* generates a log file *taxftp.log* that informs you what happened. If everything went ok, the last two lines should read:

```
Getting taxdump.tar.gz
closing connection
```

A new file should appear in the build directory called *taxdump.tar.gz*. You will have to unpack this file by typing:

```
gzip -d taxdump.tar.gz
```

```
tar -xvf taxdump.tar
```

There will be seven new files:

```
delnodes.dmp
division.dmp
gc.prt
gencode.dmp
merged.dmp
names.dmp
nodes.dmp
```

Run the importtaxdb parser to make the taxonomy data tables. *Taxdump* must be in the same directory as this parser.

```
./importtaxdb
```

*importtaxdb* has no command line parameters. *importtaxdb* generates the log file *importtaxdb\_log.txt*. If everything went ok, the output of this file should be something like:

```
Program start at Thu Sep  4 13:47:51 2003
Number of Tax ID records parsed: 191647
Number of Tax ID Name records parsed: 246263
Number of Division records parsed: 11
Number of Genetic Code records parsed: 18
```

Number of Deleted Node records parsed: 25475  
 Number of Merged Node records parsed: 4607  
 Program end at Thu Aug 12 13:49:43 2004

And records will be inserted into tables *taxdb*, *gcodedb*, *divdb*, *del* and *merge*.

## 18. Build the GODB module

Create tables of the *godb* module in the database.

```
cd $SEQH/4.godb.files
```

Make sure file *godb.sql* has line `use seqhound` close to the beginning of the file.

```
mysql -u my_id -p -P my_port -h my_server < godb.sql
```

Where *my\_id*, *my\_port* and *my\_server* should be replaced by your userid for the database, the port of the database and the IP address or the server name of the database server respectively. You will be prompted to enter your password.

This creates tables *go\_parent*, *go\_name*, *go\_reference*, *go\_synonym* in the database.

Run the *goftp.pl* script to download the gene ontology files

(<ftp://ftp.geneontology.org/pub/go/gene-associations> and  
<ftp://ftp.geneontology.org/pub/go/ontology>).

```
goftp.pl
```

There is a log file for this script called *goftp.log* that indicates that it got all of these files. Three new files should appear in the build directory called

*component.ontology*

*function.ontology*

*process.ontology*

Two other files also appear called

*gene\_association.Compugen.GenBank.gz*

*gene\_association.Compugen.UnitProt.gz*

but these are used as input files by *addgoid* in the next step.

Run the *goparser* to make the hierarchical gene ontology data tables. The three input files must be in the same directory as this parser.

```
./goparser
```

*goparser* has no command line parameters. *goparser* generates the log file *goparserlog*. If everything went ok, the output of this file should have only one NOTE line:

```
NOTE: [000.000] {goparser.c, line 101} Main: Done!
```

And records will be inserted into tables *go\_parent*, *go\_name*, *go\_reference*, *go\_synonym*.

## 19. Build the LLDB module

Create tables of the locus link module in the database.

```
cd $SEQH/5.lldb.files
```

Make sure file *ll.sql* has line `use seqhound` close to the beginning of the file.

```
mysql -u my_id -p -P my_port -h my_server < ll.sql
```

Where *my\_id*, *my\_port* and *my\_server* should be replaced by your userid for the database, the port of the database and the IP address or the server name of the database server respectively. You will be prompted to enter your password.

This creates tables *ll\_omim*, *ll\_go*, *ll\_llink*, *ll\_cdd* in the database.

Run the *llftp.pl* script to download the locus link template file (*LL\_tmpl*) which is the source for function annotation tables

(*ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL\_tmpl.gz*).

#### **llftp.pl**

This script generates the *llftp.log* file. If everything executes correctly, the last two lines of the file should read:

```
Getting LL_tmpl.gz
closing connection
```

And a new file should appear in the build directory called *LL\_tmpl.gz* which you will have to unpack using the commands

```
gzip -d LL_tmpl.gz
```

Run the *llparser* to create the set of functional annotation data tables. The input file must be in the same directory as this parser.

#### **./llparser**

*llparser* has no command line parameters. *llparser* generates the log file "*llparserlog*". At the time of writing, the output of this file will have thousands of lines like:

```
NOTE: [000.000] {ll_cb.c, line 654} LL_AppendRecord: No
NP id. Record skipped.
```

(these lines are expected since many LocusLink records are not linked to specific sequence records)

followed by the last line of the file:

```
NOTE: [000.000] {llparser.c, line 90} Main: Done!
```

Records will be inserted into tables *ll\_omim*, *ll\_go*, *ll\_llink* and *ll\_cdd*. Run the *addgoid* parser to populate the *go* annotation table. This parser uses input files that were downloaded in the previous step 13. Copy those files to this directory:

```
cp ../4.godb.files/gene_association.Compugen.GenBank.gz
./
cp ../4.godb.files/gene_association.Compugen.UniProt.gz
./
```

The files need to be unpacked.

```
gunzip gene_association.Compugen.GenBank.gz
gunzip gene_association.Compugen.UniProt.gz
```

The input files must be in the same directory as *addgoid*

```
./addgoid -i gene_association.Compugen.GenBank
after this parser has finished, use it to parse the other input file
./addgoid -i gene_association.Compugen.UniProt
```

At the time of writing, this second input file is not parsed since cross references between Swissprot and GenBank ids are not available. This is being corrected by the dbxref module project.

---

addgoid MUST BE EXECUTED AFTER ALL CORE TABLES AND LLDB TABLES HAVE BEEN BUILT; the llparser makes the ll\_go table into which the addgoid script writes. This program is dependent on tables asndb, parti, accdb and nucprot..

---

addgoid generates the log file *addgoidlog*. The output of this file will look like:

```
===== [ Sep 5, 2003 10:28 AM ]=====
ERROR: [000.000] {addgoid.c, line 235} No GI from 100K_RAT.
ERROR: [000.000] {addgoid.c, line 235} No GI from 100K_RAT.
ERROR: [000.000] {addgoid.c, line 235} No GI from 100K_RAT.
ERROR: [000.000] {addgoid.c, line 235} No GI from 100K_RAT.
```

This is normal. These errors are caused by the inability to find GI's for names of proteins/loci that are annotated in the GO input file. This problem is being addressed by the dbxref module.dir

This program writes to the existing ll\_go table that was generated by llparser.

## 20. Build the GENDB module

Change directories to the Complete Genomes directory (*comgenomes*).

**cd \$SEQH/6.comgenomes.files**

Create tables of the GENDB module in the database.

Make sure file *gendb.sql* has line `use seqhound` close to the beginning of the file.

**mysql -u my\_id -p -P my\_port -h my\_server < gendb.sql**

Where *my\_id*, *my\_port* and *my\_server* should be replaced by your userid for the database, the port of the database and the IP address or the server name of the database server respectively. You will be prompted to enter your password.

This creates table chrom in the database.

Building the GENDB module involves several steps. To simplify the process, a perl script, *comgencron\_odbc.pl* groups together all of the necessary scripts or binaries for each individual step. These scripts and binaries must be present in this directory.

They are:

*comgencron\_odbc.pl*

*shconfig.pm*

*gen\_cxx*

*pregen.pl*

*gen.pl*

*ncbi.bacteria.pl*

*genftp.pl*

*humoasn.pl*

*chrom*



*iterateparti*

*humouse\_build.sh*

*mother*

*comgen*

Before building the GENDB module, the [ *crons* ] section in configuration file *.intrezrc* should be set up properly. It should look like the following. Text in ***italics*** must be changed. Variable *mail* should have the e-mail address where you want the message to be sent to. Variable *defaultrelease* should have the release number of the GenBank files you use to build the core tables of SeqHound database (see Step 15):

```
[crons]
;customizable variables in cron jobs
;NOTE: all paths must end in '/'
pathupdates=./
pathinputfiles=./
pathinputfilescomgen=./genfiles/
mail=your_email_addr
defaultrelease=141
pathflags=./flag/
```

Make a subdirectory *flag* where the flag file *comgen\_complete.flg* will be saved.

**mkdir flag**

Run the script to build the GENDB module:

**./comgencron\_odbc.pl**

*comgencron\_odbc.pl* generates flat file *genff*, log files *bacteria.log*, *chromlog*, *comgenlog*, *gen.log*, *iteratapartilog*, a subdirectory *genfiles* and a lot of logs file with postfix *run* which will be moved to a subdirectory *logs*. It also downloads many *.asn* files which will be moved to subdirectory *genfiles*. During the process, temporary file *comff* and directory *asn* are created. They are deleted before the end of the build process. If the build process fails in the middle, they should be removed along with file *genff* manually.

There are several lines printed on the screen during the build like:

```
mail = your_email_addr
pathupdates = ./
pathinputfilescomgen = ./genfiles/
defaultrelease = 141
pathflags = ./flag/
```

```
No source or subsorce Plasmpdium falciparum NC_03043.
Update 1 chromosome type by hand.
```

It is OK to see above line.

An e-mail will be sent to the address you provide to inform if the process succeeds or fails. If everything went ok, you will see the last line in file *comgenlog* as:

```
NOTE: [000.000] {comgen.c, line 504} Main: Done.
```

The last line in file *iteratepartilog* as:

NOTE: [000.000] {iterateparti.c, line 170} Done.

The last line in file *chromlog* as:

NOTE: [000.000] {chrom.c, line 173} Done.

The last two lines in file *bacteria.log* as:

deleteing asn

See *bacteria.results* for changes to *./genff*

The last two lines in file *gen.log* as:

Removing asn

Deleting comff

*The following is a detailed explanation of the script comgencron\_odbc.pl. You may skip it.*

## 21. Generate flat file *genff*.

*genff* is a tab-delimited text file where each line in this file represents one "DNA unit" (chromosome, plasmid, extrachromosomal element etc.) belonging to a complete genome.

Column	Description
1	Taxonomy identifier for the genome
2	Unique integer identifier for a given chromosome
3	Type of molecule (1 or chromosome, 8 for plasmid, ...)
4	FTP file name for the genome without the .asn extension)
5	Full name of the organism

Here is an example of several rows from *genff*:

305	286	8	NC_003296	Ralstonia solanacearum plasmid pGMI1000MP
258594	287	1	NC_005296	Rhodopseudomonas palustris CGA009 chromosome
781	288	1	NC_003103	Rickettsia conorii chromosome
782	289	1	NC_000963	Rickettsia prowazekii chromosome
90370	290	1	NC_003198	Salmonella typhi chromosome
90370	291	8	NC_003384	Salmonella typhi plasmid pHCM1
90370	292	8	NC_003385	Salmonella typhi plasmid pHCM2
209261	293	1	NC_004631	Salmonella typhi Ty2 chromosome

The *genff* flat file is generated in two steps.

- gen.pl* which will CREATE *genff* using the eukaryotic complete genomes.
- ncbi.bacteria.pl* which will UPDATE *genff* with bacteria complete genomes.

\* both *gen.pl* and *ncbi.bacteria.pl* are dependent on *pregen.pl* so this must be in the same directory as *gen.pl* and *ncbi.bacteria.pl* when you run it.

*gen.pl* will backup the current (if it exists) *genff* as *genff.backup* and then create a new *genff* file. *gen.pl* will download asn files from NCBI's ftp site and then extract the relevant fields (as described above) and store them as records in *genff*.

The data of bacteria complete genome is written to *genff* by running *ncbi.bacteria.pl*.

This perl utility will compare the data in *genff* to the contents of the */genomes/bacteria* directory in NCBI's ftp site and then automatically update *genff*. *ncbi.bacteria.pl* will save the names of the bacteria that have been newly added to *genff* in a separate file called *bacteria.results*. You can use this file to quickly verify the results.

A sample output of *bacteria.results.pl*

```
*****PERFECT MATCH*****
Aeropyrum pernix
```

```
*****SEMI MATCHED NCBI BACTERIA*****
NCBI BACTERIA          CHROMFF
-----
Buchnera aphidicola      Buchnera sp
Buchnera aphidicola Sg    Buchnera sp
```

```
*****UNMATCHED NCBI BACTERIA*****
Agrobacterium tumefaciens C58 Cereon
Agrobacterium tumefaciens C58 UWash
```

Perfectly matched bacteria are already present in *genff*. Semi matched bacteria means that there is an organism that is closely related to a new organism. For the above example, *Buchnera aphidicola Sg* and *Buchnera aphidicola* were newly released and closely related to the *Buchnera sp*. The newly released data will have been added to *genff*. Unmatched bacteria are completely new organism and will be added to *genff*.

Both *gen.pl* and *ncbi.bacteria.pl* will create an intermediate file called *comff*, and a temporary directory *asn*. These are temporary and are critical to the functionality of the perl scripts. Both *gen.pl* and *ncbi.bacteria.pl* will delete *comff* and *asn* after execution.

While running *gen.pl* and *ncbi.bacteria.pl* you may see the following on the screen.  
No source or subsource Plasmodium falciparum NC\_03043.  
Update 1 chromosome type by hand.

It means that for the specified organism, the *asn* file is missing the chromosome type. In such a scenario, the chromosome type will default to 1 (chromosome

Once you have generated file *genff*, you will likely need to run it again periodically, in case some of the data in *genff* has changed, for example if an organism taxid changes, in which case it is crucial to rerun *gen.pl*.

Script *genftp.pl* downloads complete genome files from  
*ftp://ftp.ncbi.nih.gov/genomes/\**.

A script called *humoasn.pl* must be in the same directory as *genftp.pl* since *genftp.pl* calls the script.

---

*humoasn.pl* is a misnomer because the script actually processes files for human, mouse AND rat genomes.

---

Each of these genomes has two files called *rna.asn* and *protein.asn* (these files are called the same thing regardless of the organism that they refer to: the only way you can tell which organism the file refers to is by looking at the directory name that it came from or by looking at the contents. *genftp.pl* renames *rna.asn* and *protein.asn* files to more specific names so they can be processed with the *humoasn.pl* script.

*rna.asn* and *protein.asn* files mostly contain XM's and XP's sequences: see for example *genomes/H\_sapiens/protein*. The sequences in these files are "loose" bioseqs that have to be "stitched" together into bioseq sets by *humoasn.pl*. This allows these sequences to be processed by the mother parser in the next step.

Many new \*.asn files will appear in the *comgenomes* directory after this is run. There is no log file for this script.

## a) Populate table chrom

Binary chrom is used to populate table chrom from the list of complete genomes found in genff. Chrom generates the log file “*chromlog*”. This log will look something like:

```
===== [ Sep 5, 2003  2:30 PM ] =====
NOTE: [000.000] {chrom.c, line 130} Assigned TaxId 56636.
NOTE: [000.000] {chrom.c, line 137} Assigned Kloodge 1.
NOTE: [000.000] {chrom.c, line 144} Assigned Chromfl 1.
NOTE: [000.000] {chrom.c, line 149} Assigned Access NC_000854
NOTE: [000.000] {chrom.c, line 152} Assigned Name Aeropyrum pernix.
...
NOTE: [000.000] {chrom.c, line 167} Done.
```

## b) Delete all records from division gbchm from the tables of the core module.

This step is carried out for data integrity purpose. All the records that are inserted into the core module tables are labeled as from division gbchm. Before they are inserted, it needs to ensure no such record exists in the database. This is accomplished using binary iterateparti. iterateparti takes the division name as one parameter and deletes all GI's that are part of that division from all of the tables in the core module.

## c) Set kloodge to 0 in table taxgi

This step is also carried out for data integrity purpose. The field “kloodge” in table taxgi for all records should be set to 0 before they are updated in a later step by binary comgen.

d) Move all *Apis mellifera* related files to a subdirectory.

The chromosome, rna and protein files of *Apis mellifera* are not processed at the time of writing. They are moved to a subdirectory.

## e) Add records to the core module tables.

Since the human, mouse and rat sequences from this source (the “Complete Genomes” directory) are not a part of the GenBank release, the records are added to the core module tables by script *humouse\_build.sh*. **141**

This script feeds all chromosome, rna and protein files downloaded by *genftp.pl* to the mother parser. The mother parser makes a new division called “gbchm” (GenBank Chromosome Human and Mouse) and touches all core module tables.

Log files will be created by mother for every chromosome file processed (called *\*run*).

## f) Update field kloodge in table taxgi and field name in table accdb

Parser comgen is used to label sequences as belonging to a complete genome.

This program uses the files downloaded by *genftp.pl* and marks the complete genomes in table taxgi. This program also adds loci names into table accdb (if they are not present). comgen is dependent on the chrom table and writes to accdb and taxgi. The comgen program has to be executed after all databases are built.

Comgen writes to the log file *comgenlog* in the same directory where it is run.

## 22. Build the Strucdb module

Change to the *mmdbdata* directory.

```
cd $SEQH/7.mmdb.files
```

Create tables of the Strucdb module in the database.

Make sure file *strucdb.sql* has line use seqhound close to the beginning of the file.

```
mysql -u my_id -p -P my_port -h my_server < gendb.sql
```

Where *my\_id*, *my\_port* and *my\_server* should be replaced by your userid for the database, the port of the database and the IP address or the server name of the database server respectively. You will be prompted to enter your password.

This creates tables *mmdb*, *mmgi* and *domdb* in the database.

Make certain that the configuration files have been properly set up. These include: *.mmdbrc*, *.ncbirc* and *.intrezrc*.

In file *.mmdbrc*, variable “Gunzip” should have a value which is the path of *gunzip* on the machine (change text in *italics*). File *.mmdbrc* looks like:

```
[MMDB]
;Database and Index required when local MMDB database is used
Database = ./
Index = mmdb.idx
Gunzip = /bin/gunzip

; [VAST]
;Database required for local VAST fetches.
; Database = .
```

In file *.ncbirc*, variable *DATA* should have a value which is the path of directory *ncbi/data* on your machine. File *.ncbirc* looks like (change text in *italics*):

```
[NCBI]
ROOT=/
DATA=/my_home/compile/ncbi/data/
```

Copy file *bstdt.val* from the *ncbi/data* directory:

```
cp ~/compile/ncbi/data/bstdt.val ./
```

Run the *mmdbftp.pl* script to download the *mmdb* (Molecular Model Database) ASN.1 files from *ftp://ftp.ncbi.nih.gov/mmdb/mmdbdata*. This will take approximately 10 hours..

```
./mmdbftp.pl
```

This script writes to the *mmdb.log* file and records the files downloaded.

Approximately 20000 *\*.val.gz* files will appear in the *mmdbdata* directory after running this. Look at the first line in the *mmdb.idx* index file and this states the number of files that should have been downloaded.

Run the *cbmmdb* parser to make the MMDB and MMGI datafiles. Use:

```
./cbmmdb -n F -m F
```

This program takes about 12 hours to run and writes errors to the *cbmmdblog* file. After a typical run this file will contain:

```
===== [ Nov 3, 2003 1:21 AM ] =====
ERROR: [004.001] {cbmmdb.c, line 125} Error opening MMDB id 22339
```

```
WARNING: [011.001] {cbmmdb.c, line 240} Total elapsed time: 41857 seconds
NOTE: [000.000] {cbmmdb.c, line 245} Main: Done!
```

And records are inserted into tables *mmdb* and *mmgi*.

Run the *vastblst* parser to make the DOMDB datafile.

```
./vastblst -n F
```

This program writes errors to the *vastblstlog* file. After a typical run this file will contain no messages and records are inserted into table *domdb*

In addition, *vastblst* makes a FASTA datafile of domains called *mmdbdom.fas* in the directory where it is run.

Get the most recent *nrpdb.\** file from the NCBI ftp site in hand

```
(ftp://ftp.ncbi.nih.gov/mmdb/nrtable/nrpdb
```

Run the *pdbrp* parser to label representatives of nr chain sets in the *domdb* datatable. This parser writes to the *domdb* table. Use:

```
uncompress nrpdb*.Z
```

```
pdbrp -i nrpdb.*
```

Where *nrpdb.\** is the name of the input file set. *pdbrp* will write errors to the *pdbrplog* file in the same directory where it is run.

## 23. Build the Neighdb module

The sequence neighbours tables can be downloaded from

<ftp://ftp.blueprint.org/pub/SeqHound/NBLAST/> as MySQL database table files, as well as mysqldump output, which should be adaptable to most SQL database systems. See the readme on the ftp site for information on these files. To incorporate the mysql database table files into your instance of seqhound, simply copy the files extracted from the *nblastdb* and *blastdb* archives, downloaded from the ftp site, into your seqhound database directory in your mysql instance. To incorporate the mysql dumps of these tables into your seqhound instance, you need only pipe the contents of the dump(which are SQL statements) to your database server. In the case of mysql, simply execute:

```
gunzip -c seqhound.blastdb.SQLdump.YYYYMMDD.gz | mysql seqhound
gunzip -c seqhound.nblastdb.SQLdump.YYYYMMDD.gz | mysql seqhound
```

Be sure to fill in any required mysql options, such as username, hostname and port number.

## 24. Build the Rpsdb and Domname modules

The pre-computed rps-blast table and the domname table can be downloaded from <ftp://ftp.blueprint.org/pub/SeqHound/RPS/> as MySQL database table files, as well as mysqldump output, which should be adaptable to most SQL database systems. To incorporate the mysql database table files into your instance of seqhound, simply copy the files extracted from the *rpsdb* and *domname* archive, downloaded from the ftp site, into your seqhound database directory in your mysql instance. To incorporate the mysql dumps of these tables into your seqhound instance, you need only pipe the contents of the dump(which are SQL statements) to your database server. In the case of mysql, simply execute:

```
gunzip -c seqhound.rpsdb.SQLdump.YYYYMMDD.gz | mysql seqhound
gunzip -c seqhound.domname.SQLdump.YYYYMMDD.gz | mysql seqhound
```

Be sure to fill in any required mysql options, such as username, hostname and port number.

## 25. Build the histdb table.

```
cd $SEQH/8.hist.files  
./histparser -n F
```

This parser populates table histdb. An entry will be generated for each of the sequences that have valid accessions in table accdb that indicates that the sequence was added on this day (when you ran histparser). This parser writes to the *histparserlog*. This parser requires the accdb table and will take about 15 hours to run.

## 26. You are done with the initial build of SeqHound.

---

If you did not build any of the optional modules, you will have to remember this when setting up the *.intrezrc* configuration file for any SeqHound application.

---

Set module values to zero if you did not build them. See the following section of the *.intrezrc* configuration file.

```
example:  
[sections]  
;indicate what modules are available in SeqHound  
;1 for available, 0 for not available  
;gene ontology hierarchy (did you run goparser?)  
godb = 1  
;locus link functional annotations (did you run llparser and addgoid?)  
lldb = 1  
;taxonomy hierarchy (did you run importtaxdb?)  
taxdb = 1  
;protein sequence neighbours (did you download neighbours tables?)  
neigdb = 1  
;structural databases (did you run cbmmdb, vasttblst and pdbrep?)  
strucdb = 1  
;complete genomes tracking (did you run chrom and congen?)  
genodb = 1  
;redundant protein sequences (did you run redund?)  
redundb = 1  
;open reading frame database (currently not exported at all)  
cddb = 0  
;RPS-BLAST tables (did you download RPS-BLAST tables?)  
rpsdb = 1
```

## Catch up on SeqHound daily updates

## 27. Download all daily update files for genbank

---

Warning: There might have been a new GenBank release while you were building SeqHound, in this case you cannot get updates from *ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily-nc/* any more. You have to rebuild SeqHound with a fresh GenBank release. You should check the file *ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily/Last.Release* to make certain that it contains the same release number that was present when you started step 15.

---

```
cd $SEQH/  
mkdir seqsync
```

```

cd seqsync
ftp ftp.ncbi.nih.gov
When prompted for a name enter anonymous
When prompted for a password type myemail@home.com cd ncbi-asn1
cd daily-nc
bin
prompt
mget nc*.aso.gz
bye

```

Do not download the *con\_nc\*.aso.gz* files from this directory. SeqHound does not use them.

## 28. Download all daily update files for refseq.

From *ftp://ftp.ncbi.nih.gov/refseq/daily/* download all files past the date stamp on *gbrscu.aso.gz*. *gbrscu.aso.gz* is the latest cumulative RefSeq division which was downloaded by *asnftp.pl* and is located (in this example) in *seqhound/build/asofiles*.

```

cd $SEQH/seqsync
ftp ftp.ncbi.nih.gov
enter anonymous and your email address when prompted
cd refseq
cd daily
bin
get rsnc.****.2003.bna.Z
(where **** are files with timestamps greater than gbrscu.aso.gz)
bye

```

You must uncompress all of these files and rezip them so they can be processed by the mother parser.

```

compress -d *.Z
gzip *.bna

```

## 29. Run update and mother on all downloaded files (excluding today's one; crons will do it in the evening).

You can use the scripts *all\_update.sh* and *all\_update\_rs.sh*. You will also need mother, update and a properly configured *.intrezrc* file in the same directory as all of the daily update files.

```

cd $SEQH/seqsync
cp $COMPILE/slri/seqhound/scripts/all_update.sh .
cp $COMPILE/slri/seqhound/scripts/all_update_rs.sh .
cp $SEQH/1.core.files/.intrezrc .
cp $SEQH/1.core.files/mother .
cp $SEQH/1.core.files/update .

```

Run *all\_update.sh* first



```
./all_update.sh 141
```

where **141** is the release number.

Run *all\_update\_rs.sh* second.

```
./all_update_rs.sh 141
```

These scripts will run update and mother executables (consecutively) on all downloaded files present in the current directory.

All daily updates in SeqHound are stored in one division called gbupd regardless how long SeqHound runs without a core rebuild.

mother will make a log file called “\*run” for every file that it processes

update will make two log files called “\*gis” and “\*log” for every file that it processes

You can check that the two parsers have completed successfully. Each of the following queries should return the same number (the number of starting input files):

```
ls *aso.gz | wc -l
ls *gis | wc -l
ls nc*log | wc -l
ls nc*run | wc -l
grep Done nc*run |wc -l
```

### Setting up daily sequence updates

30. Make a new directory from where you will run daily sequence updates.

Populate this with the necessary scripts and programs.

```
cd $SEQH
mkdir updates
cd updates
cp $SLRI/seqhound/scripts/*cron_odbc.pl .
cp $SLRI/seqhound/scripts/shconfig.pm .
cp $SLRI/seqhound/build/odbc/redund .
cp $SLRI/seqhound/build/odbc/mother .
cp $SLRI/seqhound/build/odbc/update .
cp $SLRI/seqhound/build/odbc/precompute .
cp $SLRI/seqhound/build/odbc/isshoundon .
cp $SLRI/seqhound/build/odbc/importaxdb .
cp $SLRI/seqhound/build/odbc/goparser .
cp $SLRI/seqhound/build/odbc/llparser .
cp $SLRI/seqhound/build/odbc/addgoid .
cp $SLRI/seqhound/build/odbc/comgen .
```

```

cp $SLRI/seqhound/build/odbc/chrom .
cp $SLRI/seqhound/scripts/genftp.pl .
cp $SLRI/seqhound/scripts/humoasn.pl .
cp $SLRI/seqhound/scripts/humouse_build.sh .
cp $SLRI/seqhound/genomes/gen_cxx .
cp $SLRI/seqhound/genomes/pregen.pl .
cp $SLRI/seqhound/genomes/gen.pl .
cp $SLRI/seqhound/genomes/ncbi.bacteria.pl .
mkdir logs
mkdir asofiles
mkdir inputfiles
mkdir genfiles
mkdir flags

```

31. Copy the *.intrezrc* config file to the updates directory and edit it.

```

cd $SEQH/updates
cp $SLRI/seqhound/config/.intrezrc .
cp $SEQH/1.core.files/.intrezrc .

```

Text in *italics* must be changed. in [crons] section, variable *pathupdates* points to the path where the update jobs will be set up; variable *pathinputfiles* points to the path that saves the input files (other than \*.aso.gz and \*.bna.gz files from the core module and \*.asn files from the gendb module); variable *pathinputfilescomgen* points to the path that saves the input files \*.asn for the gendb module; variable *mail* indicates your e-mail address; variable *defaultrelease* is the GenBank release you build SeqHound database with; variable *pathflags* points to the path that save the flag files generated by each updating job.

```

[crons]
;customizable variables in cron jobs
;NOTE: all paths must end in '/'
pathupdates=./
pathinputfiles=./inputfiles/
pathinputfilescomgen=./genfiles/
mail=my_email
defaultrelease=141
pathflags=./flags/

```

The cron daemon may consider your home directory to be the “current directory”. For this reason, the *.intrezrc* file should be copied to your home directory too.

```

cd $SEQH/updates
cp .intrezrc ~/.

```

32. Set up the *dupdcron\_odbc.pl* cron job.

*dupdcron\_odbc.pl* (daily update cron) is a PERL script that retrieves the latest GenBank and RefSeq update files from the NCBI ftp site and then passes them to

“update” and “mother” where they are used to update the SeqHound data tables. Specifically, it

a) downloads update files with today's date (from *ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily-nc/nc\*.aso.gz* and *ftp://ftp.ncbi.nih.gov/refseq/daily/rsnc\*.bna.Z*

b) runs update

```
(update -i nc*.aso.gz)
```

and then

c) runs mother

```
(mother -i nc*.aso.gz -r version# -n F -m F -u T).
```

---

You need to know this because if you miss a few updates before setting up the cron job (and after completing the seqsync steps above) you have to run update and mother in hand using the above commands.

---

All scripts (like *dupdcron\_odbc.pl*) report success or failure via email. The mailto address is set in the *shconfig.pm* script which you have just edited.

*dupdcron\_odbc.pl* is the first cron job that has to be set up. Make a new text file called *list\_crontab* where you will list the cron jobs.

```
cd $SEQH/updates
```

```
pico list_crontabs
```

This file should have the single line

```
30 22 * * * cd /seqhound/update; LD_LIBRARY_PATH=libpath ./dupdcron_odbc.pl
```

where **libpath** should be replaced by the correct path you set up in Step 11 for environment variable LD\_LIBRARY\_PATH. You can find it out by:

```
echo $LD_LIBRARY_PATH
```

This line specifies the time to run a job on a recurring basis. It consists of 6 fields separated by spaces. The fields and allowable values are of the form:

minute (0-59) in this case 30

hour (0-23) in this case 22

day of the month (1-31) in this case \*

month (1-12) in this case \*

day-of-week (0-6 where 0 is Sunday) in this case \*

command to run

The above line indicates that *dupdcron\_odbc.pl* is to be run at 10:30 PM every day of the month, every month, regardless of the day of the week. The \* character is a wildcard. The actual command consists of changing to the directory where *dupdcron\_odbc.pl* exists (this path will have to be modified depending on your set up)

```
cd /seqhound/update;
```

and then executing the perl script

```
./dupdcron_odbc.pl
```

After adding the above line and editing it to match your setup, close the file.

To activate this crontab file, type

```
crontab list_crontabs
```

If for some reason, you want to deactivate the cron job, type:

```
crontab -r list_crontabs
```

To find out what cron jobs you have activated, type

```
crontab -l
```

For more information on setting up cron jobs on UNIX type:

```
man crontab
```

33. Set up *redundcron\_odbc.pl* to run daily.

```
cd $SEQH/updates
```

```
pico list_crontabs
```

Add the following line:

```
30 23 * * * cd /seqhound/update; LD_LIBRARY_PATH=libpath ./redundcron_odbc.pl
```

See Step 32 for the explanation of *libpath*.

After adding the above line, edit it to match your setup and close the file.

To activate this crontab file, type

```
crontab list_crontabs
```

This script basically does three things:

- a) checks if file “*nr*” is updated on the ftp site *ftp://ftp.ncbi.nlm.nih.gov/blast/db*. If it is, retrieves it
- b) drops table *redund* from the database and recreate it.
- c) rebuilds table *redund* using the downloaded *nr* file and the *redund* parser.

34. Run precompute for the first time.

First set up the configuration file

```
cd $SEQH/updates
```

```
pico .intrezrc
```

Edit the section under [precompute] to make it look like:

```
[precompute]
;precomputed taxonomy queries
MaxQueries = 0
MaxQueryTime = 10
QueryCount = 0
#path to precomputed searches has to have "/" at the end !!
path = /seqhound/precompute/
indexfile = /seqhound/precompute/index
```

Make sure the value of *path* is the absolute path of directory *precompute* you make in Step 14 and the value of *indexfile* is the value of *path* plus *index*.

Variable *path* is the directory that holds results of the precompute executable.

*indexfile* is a path to the index that will be created by precompute.

Finally, run the precompute executable:

```
cd $SEQH/updates  
./precompute -a redo
```

Where `-a redo` specifies that the program is being run for the first time.

This program basically precomputes the number of proteins and nucleic acids (and their GI values) for each taxon in the `taxgi` table. The results of this query are stored and indexed in text files (in the directory specified by `path`) if this query takes longer than `x` seconds (where `x` is defined by `MaxQueryTime` in the above `.intrezrc` file). These text files are used by SeqHound API calls such as

```
SHoundProteinsFromTaxIDIIII(taxid)
```

35. Set up *precomcron\_odbc.pl* to run daily.

```
cd $SEQH/updates  
pico list_crontabs
```

Add the following line:

```
30 1 * * * cd /seqhound/update; LD_LIBRARY_PATH=libpath ./precomcron_odbc.pl
```

See Step 32 for the explanation of `libpath`.

After adding the above line and editing it to match your setup, close the file.

To activate this crontab file, type

```
crontab list_crontabs
```

This script basically runs the command

```
precompute -a update
```

and updates the precomputed search results.

36. Set up *isshoundoncron\_odbc.pl* to run daily.

```
cd $SEQH/updates  
pico list_crontabs
```

Add the following line:

```
30 7 * * * cd /seqhound/update; LD_LIBRARY_PATH=libpath ./isshoundoncron_odbc.pl
```

See Step 32 for the explanation of “`libpath`”.

After adding the above line and editing it to match your setup, close the file.

To activate this crontab file, type

```
crontab list_crontabs
```

This script basically does two things:

- a) runs the executable called `isshoundon`. This program makes a single call to the local SeqHound API to ensure that it is working.
- b) moves all log, run and gis log files into a directory called *logs*

37. Set up *llcron\_odbc.pl* to run daily.

```
cd $SEQH/updates  
pico list_crontabs
```

Add the following line:

```
30 21 * * * cd /seqhound/update; LD_LIBRARY_PATH=libpath ./llcron_odbc.pl
```

See Step 32 for the explanation of `libpath`.

After adding the above line and editing it to match your setup, close the file.

To activate this crontab file, type

```
crontab list_crontabs
```

This script basically repeats the actions listed in step 14 above and re-creates the locus link tables in SeqHound. This includes:

- a) getting the latest *LL\_tmpl.gz* file from the NCBI ftp site.
- b) removing the locus link tables from SeqHound
- c) running *llparser*
- d) getting 2 GO annotation files from GO ftp site
- e) running the *addgoid* parser on these two files

38. Set up *comgencron\_odbc.pl* to run daily.

```
cd $SEQH/updates
```

```
pico list_crontabs
```

Add the following line:

```
30 21 * * * cd /seqhound/update; LD_LIBRARY_PATH=libpath ./comgencron_odbc.pl
```

See Step 32 for the explanation of `libpath`.

After adding the above line and editing it to match your setup, close the file.

To activate this crontab file, type

```
crontab list_crontabs
```

This script basically repeats the actions listed in step 15 above and re-creates the chrom table in SeqHound and updates the complete genome information in the core tables. This includes:

- a) generating a list of “DNA units” that belongs to a complete genome,
- b) downloading complete genome files from NCBI ftp site,
- c) rebuilding table *chrom*
- d) removing all records in the core tables that belongs to division “gbchm”,
- e) running script *humous\_build.sh* to insert records into core tables
- f) resetting the *kloodge* field in table *taxgi* for all records to 0
- g) updating *kloodge* by running parser *comgen*

## Setting up SeqHound servers. Overview

### 39. Setting up SeqHound servers. Overview.

There are two web server applications that make up the SeqHound system:

- a) `wwwseekgi` produces html pages for the SeqHound web interface and
- b) `seqrem` processes requests to the SeqHound remote API.

Step 40 shows you how to find the two directories where you will set up these two applications (assuming that you are using a default installation of Apache). The two directories are called:

*cgi-bin*

*htdocs*

Step 40 may be skipped if you already know or have already been told where these two directories are.

Steps 41 - describe the files that must be placed into these two sub-directories in order to start the `wwwseekgi` and `seqrem` servers.

### 40. Examining the *httpd.conf* file for Apache.

These instructions assume that you already have an Apache server running. In order to proceed further you must locate the directory where executables will be run from (called "*cgi-bin*" in a default set-up of Apache) and a directory that contains html documents (called "*htdocs*" in a default set-up of Apache). You can find (and reset) the location of these two directories in an Apache configuration file called "*httpd.conf*". In a default set-up of Apache, the *httpd.conf* file can be accessed by changing to the directory:

```
cd /etc/apache
```

and then opening the *httpd.conf* file found in this directory using a text editor such as `pico`:

```
pico httpd.conf
```

To find the *cgi-bin* directory location, look for the line beginning with "ScriptAlias". In the default set-up, this line looks like this:

```
ScriptAlias /cgi-bin/ "/var/apache/cgi-bin/"
```

In this example, the path to the *cgi-bin* directory is `/var/apache/cgi-bin/`. Write this path down, whatever it is.

To find the *htdocs* directory, look for the line beginning with "DocumentRoot". In the default set-up, this line looks like this:

```
DocumentRoot "/var/apache/htdocs/"
```

In this example, the path to the *htdocs* directory is `/var/apache/htdocs/`. Write this path down, whatever it is.

Also make a note of the line beginning with "User" and "Group" (who has ownership to the server). In a default Apache set-up, these lines are likely

```
User nobody
```

Group nobody

Make a note of this, whatever it is.

Exit from the *httpd.conf* file and save your changes. If you made changes to the file, you must restart the Apache server using the command:

```
/usr/apache/bin/apachectl restart
```

See the *Trouble Shooting* section at the end for more information on this.

In the steps below you will set up the SeqHound server by adding to these two directories

Contents of the <i>cgi-bin</i> and <i>htdocs</i> directories	
directory	contents
<i>cgi-bin</i>	the SeqHound <i>wwwseekgi</i> and <i>seqrem</i> server applications will placed here
<i>htdocs</i>	all of the static html pages used by the SeqHound interface will be placed here

#### 41. Set up the *cgi-bin* directory.

Move to the *cgi-bin* directory you found in the step above. For the default set-up:

```
cd /var/apache/cgi-bin/
```

make a new subdirectory here called SeqHound

```
mkdir seqhound
```

```
cd seqhound
```

copy the SeqHound server applications here:

```
cp $COMPILE/slri/seqhound/build/odbc/seqrem .
```

```
cp $COMPILE/slri/seqhound/build/odbc/wwwseekgi .
```

also copy the following files to this directory:

```
cp $COMPILE/slri/seqhound/html/seekhead.txt .
```

```
cp $COMPILE/slri/seqhound/html/seektail.txt .
```

```
cp $COMPILE/slri/seqhound/html/seekhead.txt pics/.
```

```
cp $COMPILE/slri/seqhound/config/.intrezrc .
```

```
cp $COMPILE/slri/seqhound/config/.ncbirc .
```

#### 42. Edit the *.ncbirc* configuration file.

Open the file with a text editor such as pico.

The setting for Data should contain a path to the *ncbi/data* directory. This directory was downloaded as part of the ncbi toolkit in step 2.

```
-----example .ncbirc file begins-----
[NCBI]
Data=/home/ncbi/data
-----example .ncbirc file ends-----
```

#### 43. Edit the *.intrezrc* configuration file.

Refer to step 14 in the current section for setting up of the *.intrezrc* file. The settings for username, password, dsn and database in section [datab] should be



valid for the SeqHound database you have just built, and the setting for `path` and `indexfile` in section `[precompute]` should point to the valid path as in step 34 in the current section. Set up the *index.html* file for the web interface.

Move to the *htdocs* directory for your web-server. In the default case:

```
cd /var/apache/htdocs/
```

Make a SeqHound directory here:

```
mkdir seqhound
```

```
cd seqhound
```

Copy the *index.html* page to this directory:

```
cp $COMPILE/slri/seqhound/html/index.html .
```

Open the file in a text editor like *pico* and edit it so that its action points to the *wwwseekgi* server.

```
pico index.html
```

then edit the line

```
<FORM ACTION="/cgi-bin/seqhound/wwwseekgi" METHOD="GET">
```

where `"/cgi-bin/seqhound/wwwseekgi"` should specify the path to the *wwwseekgi* executable.

#### 44. Set up ODBC configuration file *.odbc.ini*:

Move to the home directory of the owner of the binary *seqrem* in directory */var/apache/cgi-bin/*. Text in *italics* should be changed (see Step 10):

```
cd /homedir
```

Set up file *.odbc.ini* as the following (text in *italics* should be changed):

```
[mysqlsh]
Description = MySQL ODBC 3.51 Driver DSN
Trace       = On
TraceFile   = stderr
Driver      = /software/64/unixodbc/odbc/lib/libmyodbc3.so
DSN        = mysqlsh
SERVER     = my_server
PORT       = my_port
USER       = user_id
PASSWORD   = my_pwd
DATABASE    = seqhound
```

#### 45. Set permissions on the *cgi-bin* directory.

Move to the *cgi-bin* directory.

```
cd /var/apache/cgi-bin/
```

Change the user and group ownership to *nobody* (or whatever the values of “User” and “Group” were set to in step 40).

```
chown -R nobody:nobody seqhound
```

#### 46. Set permissions on the *htdocs* directory.

Move to the *htdocs* directory.

```
cd /var/apache/htdocs/
```

Change the user and group ownership to `nobody` (or whatever the values of “User” and “Group” were set to in step 40).

```
chown -R nobody:nobody seqhound
```

47. Test the SeqHound web interface.

Open an internet browser and, in this example, go to the url

```
http://yourmachinename/cgi-bin/seqhound/
```

You should see the front page of the SeqHound `wwwseekgi` interface.

## Trouble-shooting notes

### Error logs

Error logs for each of the SeqHound parsers are described in the steps above where the parser is used to initially build a given SeqHound module.

Error logs for the SeqHound *wwwseekgi* server software is located in the same directory as the executable; see *wwwseekgilog*.

Error logs for the *seqrem* server are located in the same directory as the executable; see *seqremlogs*.

Error logs for the Apache server software are located (in a default set-up) in */var/apache/logs*.

### Recompiling SeqHound

If you make changes to and recompile SeqHound, you should first do a clean of the existing object files and executables. If you are still in a super-user shell, you may wish to exit this shell and return to the shell where you had set all of your environment variables. These variables are required by the clean and make scripts.

COMPILE	compile directory
SLRI	slri directory
NCBI	ncbi directory
CC	gcc
PATH	/usr/local/bin:/usr/ccs/bin:\${PATH}
EXTRAOPT	-D_FILE_OFFSET_BITS=64
ODBC	(path to unixodbc)
LD_LIBRARY_PATH	(path to mysql and odbc libraries)

The first three variables refer to directories that are created during the above instructions.

To clean, run any make file with the 'clean' target. For example, to clean the cgi executables, type:

```
cd $COMPILE/slri/seqhound/cgi
make -f make.seqrem clean
make -f make.wwwseekgi clean
```

### Restarting the Apache server

If changes are made to the *httpd.conf* file, the Apache server must be restarted for the changes to take effect. Use the *apachectl* script to do this

For a default install of Apache this script is in the directory

```
cd /usr/apache/bin
```

and to start the server, type

```
./apachectl restart
```

or to get a list of *apachectl* script commands type

## **./apachectl**

### **Other useful links**

SLRI on Sourceforge	<a href="http://sourceforge.net/projects/slrtools/">http://sourceforge.net/projects/slrtools/</a>
NCBI Info Engineering branch:	<a href="http://www.ncbi.nlm.nih.gov/IEB/">http://www.ncbi.nlm.nih.gov/IEB/</a>
Concurrent Versioning System (CVS)	<a href="http://www.cvshome.org/">http://www.cvshome.org/</a>
MySQL	<a href="http://www.mysql.com/">http://www.mysql.com/</a>

### **Parser schedule**

Parsers are run on a periodic basis as “cron” jobs on Unix platforms and as “Schedules tasks” on Windows platforms.

The cron job schedule is set up in the file “*seqhound/update/list\_crontabs*” on UNIX platforms

The setup for the current production version of SeqHound described in these instructions is shown below:

```
0 19 * * * /arena/seqhound/update/llcron_odbc.pl
0 21 * * * /arena/seqhound/update/redundcron_odbc.pl
30 22 * * * /arena/seqhound/update/dupdcron_odbc.pl
30 24 * * * /arena/seqhound/update/precomcron_odbc.pl
0 7 * * * /arena/seqhound/update/issboundoncron_odbc.pl
```

### **MySQL errors**

ERROR 1153 at line X: Got a packet bigger than 'max\_allowed\_packet'

When MySQL receives a packet bigger than `max_allowed_packet` bytes, it issues a Packet too large error and closes the connection. For example, when a single SQL statement from a mysqldump being imported exceeds the value for "max\_allowed\_packet" configured on the MySQL server. Increasing this value to 64MB from the default 16MB should resolve the error. This value may be changed in the global config or setting this on the running server via:

```
set global max_allowed_packet=67108864;
```

Please see [http://dev.mysql.com/doc/mysql/en/Package\\_too\\_large.html](http://dev.mysql.com/doc/mysql/en/Package_too_large.html) for more information

## 5. Description of the SeqHound parsers and data tables by module

### What are modules?

The SeqHound system is divided into one required “core” module and several optionally configured modules. Modules are groups of tables and API calls that are filled using a common data resource, for example the 3D structures. The purpose of this division is to give the user an option to control hardware resources and complexity of system administration when parts of the SeqHound system are not required. The list of SeqHound modules and their data resources is contained in the table below.

After a system build, the module information is recorded in the configuration file which is then utilized by the API to determine if certain operations can be achieved with the current setup. The configuration file is called *.intrezrc* (Unix platforms) The relevant section of this file is under the heading “[sections]”. Consult [Section 4](#) under “*Building the SeqHound system*” for more information on specifying the available modules in this file.

### How to use this section.

This section describes the SeqHound system in detail module by module.

Parser and data tables associated with a given module are described under the section for that module.

A brief description of all of the parsers can be found below and in the Table 2 of the *SeqHound paper*. The table is repeated below and will be updated here.

---

Note: It is assumed that you have read the material in [section 3](#) and [section 4](#) before delving into this material. These two sections include everything you need to know to start using the SeqHound remote API or to install you own local version of SeqHound. This section is intended for users who may want more details about how SeqHound is constructed and exactly what it is doing behind the scenes. This section is also meant as background material for developers who want to further develop the SeqHound system

---

### Parser descriptions

Parser descriptions contain the following headings and information:

<b>purpose:</b>	a brief description of what the parser is for
<b>logic:</b>	more details on the parser – see update parser for example
<b>module:</b>	what module the parser belongs to
<b>input files:</b>	input files required by the parser (also available in Table 2)
<b>tables altered:</b>	tables in SeqHound that are modified or created by the parser (also available in Table 2)
<b>source code location:</b>	location of the parser source code in the slri development tree
<b>config file dependencies:</b>	what configuration file parameters must be set for the parser to work
<b>command line parameters:</b>	used by the parser
<b>example use:</b>	of the parser from the command line
<b>associated scripts:</b>	that are used to run the parser
<b>error and run-time logs:</b>	where they are located and what is in them
<b>troubleshooting:</b>	problems that may occur with this parser
<b>additional info:</b>	where to find it

## Table descriptions

Each table that is relevant to a module is described under that module.

Here is an example with comments in brackets. A data table description consists of the following sections and content:

<b>Database:</b>	that the table belongs to (almost always SeqHound)
<b>Table:</b>	name of the table like “accdb”

**Definition:** a brief description of the table's purpose (for example, "This table correlates gi's to accession identifiers).

**Observation:** notes about the table

**Source db:** where does this table's information come from

**Source file:** the source file (used by the parser to fill this table) location is listed here

**Parser:** the name of parser if a single parser is responsible for filling this entire table.

This is followed by a summary of the table's definition (for example):

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
gi	int(11)	No	0	GenInfo Identifier
asn1	longblob	Yes	NULL	BioSeqs: Sequences

and indexes

Keyname	Type	Field
PRIMARY	PRIMARY	gi
iasndb_rowid	INDEX	rowid
iasndb_gi	INDEX	gi

(each field is then described...only one field description is shown for this example)

\*\*\*gi\*\*\*

**description:** definition of the column (for example, "GenInfo sequence record identifier")

**example:** of a column entry (for example, "1232452")

**default value:** if the value has a default value, it is listed here

**ASN.1 structure:** If the value in this column is derived from an NCBI data structure, this gives you a quick idea of where to locate it--you can find more info by searching

for this data structure at:

*<http://www.ncbi.nlm.nih.gov/IEB/ToolBox/SB/hbr.html>.*

for example, “Bioseq->Seq-id (choice 12)

Alternatively, the column may store a binary object that is an NCBI or SLRI ASN.1 object. Descriptions of SLRI ASN.1 objects may be found at

*<http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slrtools/slri/seqhound/asn>.*

**source:**

If the value in this column is derived from a text file, this describes how to find the information present in this column in the source file (for example in the fourth column)

**parser:**

this is the parser(s) that retrieves the value from some other db (for example, mother)

**function:**

this is the parser function that retrieves the value

**API:**

if a SHound API function retrieves this value from this table it is listed here (for example, `ShoundFindAcc`)

**more info:**

(other notes)



**An overview of the SeqHound data table structure**

An overview of the SeqHound data table structure is available as a separate document in pdf format. See [http://www.blueprint.org/seqhound/api\\_help/docs/SeqHound\\_Schema\\_Prod.pdf](http://www.blueprint.org/seqhound/api_help/docs/SeqHound_Schema_Prod.pdf).

## Parsers and resource files needed to build and update modules of SeqHound.

This table will be updated shortly

Input File	Resource	Parser	Tables Modified	Module
ASN.1 sequences	<a href="ftp://ftp.ncbi.nih.gov/ncbi-asn1/*.aso">ftp://ftp.ncbi.nih.gov/ncbi-asn1/*.aso</a> <a href="ftp://ftp.ncbi.nih.gov/refseq/cumulative/*.bna">ftp://ftp.ncbi.nih.gov/refseq/cumulative/*.bna</a>	mother	asndb, parti, nucprot, accdb, pubseq, taxgi, sendb, sengi	core
ASN.1 sequences	<a href="ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily-nc/*.aso">ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily-nc/*.aso</a> <a href="ftp://ftp.ncbi.nih.gov/refseq/daily/*.bna">ftp://ftp.ncbi.nih.gov/refseq/daily/*.bna</a>	update	asndb, parti, nucprot, accdb, pubseq, taxgi, sendb, sengi	core
FASTA nr database	<a href="ftp://ftp.ncbi.nih.gov/blast/db/nr">ftp://ftp.ncbi.nih.gov/blast/db/nr</a>	redund	redund	redundb
List of complete genomes (flat file)	<a href="http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slrtools/slri/seqhound/genomes/chromff">http://cvs.sourceforge.net/cgi-bin/viewcvs.cgi/slrtools/slri/seqhound/genomes/chromff</a>	chrom	chrom	gendb
ASN.1 for complete genomes	<a href="ftp://ftp.ncbi.nih.gov/genomes/*.asn">ftp://ftp.ncbi.nih.gov/genomes/*.asn</a>	comgen	taxgi, accdb	gendb
Taxonomy release (flat file)	<a href="ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar">ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar</a>	importtaxdb	TAX, GCODE, DIV, del, merge	taxdb
ASN.1 MMDB release	<a href="ftp://ftp.ncbi.nih.gov/mmdb/mmdbdata/*.val">ftp://ftp.ncbi.nih.gov/mmdb/mmdbdata/*.val</a>	cbmmdb	mmdb, mmgi	strucdb

MMDB (database table)	mmdb table	vastblst	domdb	strucdb
3-D chain BLAST sets (flat file)	<i>ftp://ftp.ncbi.nih.gov/mmdb/nrtable/nr pdb.*</i>	pdbrep	domdb	strucdb
FASTA nr database	<i>ftp://ftp.ncbi.nih.gov/blast/db/nr</i>	nblast	nrB	neigdb
BLAST ASN.1 results	nrB table available nrB and nrN tables available at <i>ftp://ftp.blueprint.org/pub/SeqHound/ NBLAST/</i>	nbraccess	nrN	neigdb
LL_tmpl (flat file)	<i>ftp://ftp.ncbi.nih.gov/refseq/LocusLink/ LL_tmpl</i>	llparser	ll_omim, ll_go, ll_llink, ll_cdd	lldb
gene_associaton .com				
pugen.GenBank /Swissprot (flat files)	<i>http://www.geneontology.org</i>	addgoid	ll_go	lldb
function.ontolog y			go_parent, go_name, go_reference, go_synonym	
process.ontolog y	<i>http://www.geneontology.org</i>	goparser		godb
component.onto logy (flat files)				
CDD database	<i>ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/</i>	domname	domname	rpsdb

FASTA nr database and CDD database	<i>ftp://ftp.ncbi.nih.gov/blast/db/nr;</i> <i>ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/</i> DOMNAME and RPSDB tables available at <i>ftp://ftp.blueprint.org/pub/SeqHound/</i> <i>RPS/</i>	rpsdb	rpsdb	rpsdb
--	--	-------	-------	-------

**core module****mother parser**

Last updated September 30, 2004

**purpose:**

The mother parser is the first parser that is used to initially build the SeqHound “core” set of data tables. The input files consist of the latest release of GenBank and RefSeq in binary ASN.1 format available on their ftp site. The resulting SeqHound data tables hold DNA, RNA and protein sequence record information.

The mother parser is also used in conjunction with histparser, update parser and GenBank daily updates to update SeqHound so that that sequence information is synchronized with that of GenBank.

**module:** core

**input files:**

latest GenBank release (*ftp://ftp.ncbi.nih.gov/ncbi-asn1/\*.aso*)

latest RefSeq release (*ftp://ftp.ncbi.nih.gov/refseq/cumulative/\*.bna*)

daily GenBank release (*ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily-nc/\*.aso*)

daily RefSeq release (*ftp://ftp.ncbi.nih.gov/refseq/daily/\*.bna*)

**tables altered:**

asndb, parti, nucprot, accdb, pubseq, taxgi, sendb, sengi, bioentity, bioname

**source code location:**

*slri/seqhound/parsers/mother.c*

**config file dependencies:**

The relevant configuration file is:

*slri/seqhound/config/.intrezrc* (for Unix platforms) or

The relevant section of the configuration file is:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_.odbc.ini_file
database=seqhound
local=
```

Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.)

### command line parameters:

Typing “**./mother -**” at the command line while in the directory where mother resides will return a list of command line parameters and default settings.

---

Note that **-n** and **-m** are listed for historical Codebase purposes. These values are always **F** for ODBC.

---

For example:

```
> ./mother -
mother arguments:
-i Filename for asn.1 input [File In]
-r Release [String]
-n Initialize the ASNDB database file [T/F] Optional
  default = F
-m Initialize the remaing database files [T/F] Optional
  default = F
-u Is this an update [T/F] Optional
  default = F
-c Is this a file for human/mouse complete genome [T/F] Optional
  default = F
-t Read input file in text mode [T/F] Optional
  default = F
```

### example use:

```
mother -i nc1227.aso -r 135 -n F -m F
```

Note that mother is normally run under the control of a script (see below).

### associated scripts:

The initial build of SeqHound is executed using the script called “*seqhound\_build.sh*”

See “*slri/seqhound/scripts/seqhound\_build.sh*”

This script cycles through all the GenBank and RefSeq release files downloaded by *asn\_fip.pl* unzips them, processes them with mother and then zips them up again.

The script must be run in the directory containing the GenBank and RefSeq release files and a copy of the mother parser. The script takes one argument (a release number).

```
./seqhound_build.sh 135
```

mother is also called by the scripts that generate the daily updates (see “associated scripts” under update parser).

#### **error and run-time logs:**

mother writes to a log file called “*rsnc0506run*” where rsnc0506 refers to the GenBank release or update file that was being processed. The log files are created in the directory *logs*.

#### **troubleshooting:**

#### **additional info:**

See readme files that accompany the input files on the GenBank ftp site.

See data table descriptions for each of the tables that are listed under “tables altered”

**update parser**

Last updated September 30, 2004

**purpose:**

The update parser is used in conjunction with the mother parser and daily update files from GenBank to keep an up to date set of sequence records in SeqHound that accurately reflects the current contents of GenBank.

**update logic:**

The update and mother parsers are run on a daily basis. Both take the same set of two input files. One input file is the daily update from GenBank and the other is the daily update from the RefSeq database.

---

Note that the first time the core tables are built histparser must be run before update.

---

Each of these files is first passed to update and then to mother (this order of operations is very important).

For each bioseq in the input file, update retrieves the GI and accession number.

update then looks for this pair of GI and accession identifiers in the SeqHound accdb table.

If neither the accession (nor the GI) is found, then the record in the daily update represents a GI that is to be ADDED to SeqHound.

If the accession in the update file is found in SeqHound and is associated with the same GI (as listed in the update file) then the record in the daily update represents a GI that has been CHANGED. This means that the sequence record was resubmitted to GenBank with the same GI; the sequence remains the same but the associated annotation has changed.

If the accession in the update file is found in SeqHound but the GI associated with this accession differs between the update file and SeqHound, then the GI that is newly associated with the accession represents a change in the sequence. The accession and updated GI pair point to a sequence record that will be ADDED to SeqHound. The accession and old GI (currently in SeqHound) point to a record in SeqHound that will be KILLED (deleted).

update then records in the SeqHound history table whether the GI and accession in the update file represent an ADDED or CHANGED record. Update also records in the history table those sequence records currently in SeqHound that will be KILLED (see above).

update completes this process for every bioseq in the update file.



A list of ADDED, CHANGED and KILLED GI's is written to the file *rsnc0506gis* where *rsnc0506* refers to the update file that was being processed.

update then deletes from the following tables: *parti*, *accdb*, *sendb*, *sengi*, *taxgi*, *pubseq*, *nucprot* AND *asndb*.

update writes errors and messages to a log file called *rsnc0506log* where *rsnc0506* refers to the update file that was being processed.

The update file is then passed to *mother* as an argument along with another Boolean argument that indicates that the file being parsed is a daily update. *mother* processes this file as described under the *mother* parser description with one exception; the division (see the tables *parti*, *asndb* and *sendb*) is recorded as “gbupd” for all sequences present in the update file that are NEW or CHANGED.

*mother* writes to a log file called *rsnc0506run* where *rsnc0506* refers to the update file that was being processed.

**module:** core

**input files:**

There are two input files.

1) daily updates from GenBank

(*ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily-nc/nc0506.aso.gz*) where 0506 means May 6<sup>th</sup>

2) daily updates for RefSeq (*ftp://ftp.ncbi.nih.gov/refseq/daily/rsnc.0506.2004.bna.gz*) where 0506 means May 6<sup>th</sup>).

**tables altered:**

*histdb*, *parti*, *accdb*, *sendb*, *sengi*, *taxgi*, *pubseq*, *nucprot*, *asndb*, *bioentity*, *bioname*.

**source code location:**

*slri/seqhound/updates/update.c*

**config file dependencies:**

The relevant configuration file is:

*slri/seqhound/config/.intrezrc* (for Unix platforms)

The relevant section of the configuration file is:

[datab]

```
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_.odbc.ini_file
database=seqhound
local=
```

Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.)

### command line parameters:

Typing “**./update -**” at the command line while in the directory where update resides will return a list of command line parameters and default settings. For example:

```
> update -
update  arguments:
-i  ASN.1 input file [File In]
```

---

Note that update is normally run under the control of a script (see below).

---

### example use:

```
update -i nc0506.aso
```

### associated scripts:

The daily update process is controlled by the “*dupdcron\_odbc.pl*” script. This script gets the daily update files and passes them to the update and mother parsers.

Two other shell scripts are used to process a batch of GenBank update files or a set of RefSeq update files; these are, respectively:

```
all_update.sh
```

```
all_update_rs.sh
```

These scripts are useful for updating when a number of daily updates have been missed.

**error and run-time logs:**

Update writes to a log file called “*rsnc0506run*” where rsnc0506 refers to the GenBank release or update file that was being processed. The log file is created in the directory *logs*.

**troubleshooting:****additional info:**

See readme files that accompany the input files on the GenBank ftp site.

See data table descriptions for each of the tables that are listed under “tables altered”

**asndb table**

Last updated August 5, 2004

**Database:** SeqHound

**Module:** core

**Table:** asndb

**Definition:** Central table that stores sequence records in binary ASN.1 format and indexed by gi.

**MySQL**

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
gi	int(11)	No	0	GenInfo Identifier
asn1	longblob	Yes	NULL	BioSeqs: Sequences
division	varchar(25)	No		
release	varchar(10)	No		
type	varchar(15)	No		

**MySQL Indexes**

Keyname	Type	Field
PRIMARY	PRIMARY	gi
iasndb_rowid	INDEX	rowid
iasndb_gi	INDEX	gi

**Observation:**

**\*\*\*gi\*\*\***

**description:** GenInfo primary sequence record identifier  
**example:** 4001823  
**default value**  
**ASN.1 structure:** Bioseq->seqid->gi  
**parser:** mother  
**function:** FillASNDB calls GetGI  
**more info:**

**\*\*\*asn.1\*\*\***

**description:** bioseq in binary asn.1  
**example:** An example of a bioseq is included in the Appendix.  
**default value** n/a  
**ASN.1 structure:** Bioseq->seqid->gi  
**parser:** mother  
**function:** ToBioseqSEQENTRY  
**API:** SHoundSequenceLength  
**more info:**

**\*\*\*division\*\*\***

**description:** division name is the name of the asn.1 file taken as input by the parser  
**example:**  
**default value**

**ASN.1 structure:** n/a  
**parser:** mother  
**function:** FileNameFind gets the name of the ASN.1 input file that the parser is processing

**more info:**

\*\*\*release\*\*\*

**description:** release is the name of the release given as a command line input to the mother parser

**example:** 135

**default value** n/a

**ASN.1 structure:** n/a

**parser:** mother

**function:** n/a

**more info:**

\*\*\*type\*\*\*

**description:**

**example:**

**default value**

**ASN.1 structure:**

**parser:**

**function:**

**API:** SHoundMoleculeType

**more info:**

**parti table**

Last updated August 5, 2004

---

Note: This table is documented for historical purposes and may be deprecated in a future release.

---

**Database:** seqhound

**Table:** parti

**Module:** core

**Definition:** This table maps the gi to the database partition that the gi is stored in.  
MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
gi	int(11)	No	0	DNA GenInfo Identifier
division	char(15)	Yes	NULL	

MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	gi
iparti_rowid	INDEX	rowid
iparti_gi	INDEX	gi
iparti_div	INDEX	division

**Observation:** The parti database acts as an index of the asndb data files.  
The parti database maps a gi to a particular division of the asndb database. The database documentation implies that there is one particular database called asndb. However this is not the case in CodeBase. At one point, due to memory constraints in CodeBase



(CodeBase cannot have file size > 2.0 GB), it was not possible to store all the ASN information in one file. Instead, the information that should be stored in one asndb file was partitioned into divisions (based on the filename from which mother parses the information) and stored into numerous files that collectively represent the asndb database.

For example, suppose a request is made for a particular ASN structure given a gi. That gi is looked up in the parti database to determine which partition file the ASN information is stored in. Once the partition is obtained, the ASN.1 information is then extracted from that file. Thus the parti database is an auxiliary database that does not provide any public information. eg. gi 21676384 is parsed from *gbest167.aso.gz*, the corresponding record in *PARTI.dbf* is gi: 21676384 division:gbest167 then in *gbest167.dbf* is the ASN.1 record for gi 21676384

The exception to this is for gis that have been updated (present in the daily update files, see update parser description), in which case the division is always 'gbupd'. If the backend database is not CodeBase, the parti database is not used because the memory limitation is not a factor. At present, both the parti database and the asndb database get created and populated, but in the near future, it is expected that the parti database will be removed since it is redundant.

**Source org:** NCBI  
**Source file:** \*.aso.gz from ftp://ftp.ncbi.nih.gov/ncbi-asn1/  
**FTP script:** asnftp.pl  
**Parser:** mother

\*\*\*gi\*\*\*

**description:** GenInfo Identifier for a nucleic acid sequence record

**example:** 21676275  
**default value:** n/a  
**ASN.1 struct:** Bioseq->Seqid->gi (choice 12)  
**parser:** mother  
**function:** FillASNDDB, AppendRecordPARTI  
**API:** the gi information is never used externally. It serves as an index to the collection of files that make up the asndb database.

\*\*\* division \*\*\*

**description:** which division of the database was the gi parsed from (see Observation above).  
**example:** gbest167  
**default value:** n/a  
**source:** extracted from the name of the data file: eg *gbest167.aso.gz* yields gbest167  
**parser:** mother  
**function:** n/a, the division is not extracted from any particular field, it is extracted from the name of the file from which the ASN information is extracted, which is provided as a command line argument.  
**API:** the division information is never used externally. It servers as an index to the collection of files that make up the asndb database.

**nucprot table**

Last updated August 5, 2004

**Database:** seqhound  
**Table:** nucprot  
**Module:** core  
**Definition:** This table maps the gi of a protein to the gi of a DNA.

## MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
gin	int(11)	No	0	DNA GenInfo Identifier
gia	int(11)	No	0	Protein GenInfo Identifier

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	gia
inucprot_rowid	INDEX	rowid
inuc_gin	INDEX	gin
inuc_gia	INDEX	gia

**Observation:** A DNA may map to more than one protein. A protein can only map to one DNA. Not every DNA may map to a protein, eg synthetic DNA fragments typically do not have a protein. In a bioseqset structure, the first bioseq record is a nucleic acid, subsequent bioseq records correspond to proteins. In such a case, the entries added to the nucprot database would be:

record 1) gi\_1 gi\_2

record 2) gi\_1 gi\_3

record 3) gi\_1 gi\_4

Each record would be a mapping of the nucleic acid (gi\_1) to each protein (gi\_2 ... gi\_N)

**Source org:** NCBI

**Source file:** *\*.aso.gz from ftp://ftp.ncbi.nih.gov/ncbi-asn1/*

**FTP script:** *asnftp.pl*

**Parser:** mother

### \*\*\*gin\*\*\*

**description:** GenInfo Identifier for a nucleic acid sequence record

**example:** 27464927

**default value:** n/a

**ASN.1 struct:** Bioseq->Seqid->gi (choice 12)

**parser:** mother

**function:** FillNUCPROT

**API:** SHoundDNAFromProtein returns a DNA gi given a Protein gi.

### \*\*\*gia\*\*\*

**description:** GenInfo Identifier for a protein sequence record

**example:** 27464928

**default value:** n/a

**ASN.1 struct:** Bioseq->Seqid->gi (choice 12)

**parser:** mother  
**function:** FillNUCPROT  
**API:** SHoundProteinFromDNA returns a Protein gi given a DNA gi.

**accdb table**

Last updated August 5, 2004

**Database:** seqhound

**Table:** accdb

**Definition:** This table maps gi's to accession numbers

MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
gi	int(11)	No	0	GenInfo Identifier
db	varchar(15)	No		
name	varchar(30)	No		
namelow	varchar(30)	Yes	NULL	
access	varchar(20)	No		
chain	varchar(20)	Yes	NULL	
release	varchar(20)	Yes	NULL	
version	int(11)	Yes	NULL	
title	text	Yes	NULL	

MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	gi db access
iaccdb_rowid	INDEX	rowid
iaccdb_gi	INDEX	gi

iacddb_db	INDEX	db
iacddb_name	INDEX	name
iacddb_namelow	INDEX	namelow
iacddb_acc	INDEX	access

**Observation.:****\*\*\*gi\*\*\***

**description:** primary sequence record identifier  
**example:** 2313082  
**default value:** ?  
**ASN.1 structure:** Bioseq-->Seqid (choice 12)  
**parser:** mother, update  
**function:** GetGI  
**API:** SHoundFindAcc returns a gi given an accession.  
**more info:** gi is not unique in accdb In some cases one GI has different names attached to it and there are more entries in **accdb** for that GI.

**\*\*\*db\*\*\***

**description:** This is the source database of the sequence record.  
**Example:** gb means GenBank  
embl means EMBL  
pir means Protein Information Resource (PIR)

sp means Swiss-Prot  
pbs means other database  
ref means RefSeq  
dbj means DNA Database of Japan (DDBJ)  
prf means PRF  
pdb means Protein Data Bank (PDB)  
other means some other database

**ASN.1 struct:** Bioseq-->Seqid (depending on the choice, one of the db's is chosen)

**parser:** mother, update

**function:** FillAccDB

**API:** There is no way to directly retrieve the value of db from this table using the API, but you can use SHoundSeqIdFromGi which returns a SeqId structure given a gi. Note that names of 'other' db's will not appear in this column (like TIGR) since this is in Bioseq->Seq-id->Dbtag->db (a string)

\*\*\*name\*\*\*

**description:** This is an accession number from a foreign db.

**Example entry:** AF060234

**ASN.1 struct:** For all db's except PDB and 'other':  
Bioseq->Seq-id->Textseq-id->name  
For PDB sequences:  
Bioseq->Seq-id->PDB->seq-id->PDB-mol-id (4 characters)  
For other databases:  
Bioseq->Seq-id->Dbtag->Object-id->string or integer

**parser:** mother, update



**function:** WriteACCDB  
**API:** SHoundFindName searches on this field and returns a GI  
SHoundGetNameByGi  
**more info:** Names in the name field are unique in combination with db

\*\*\*access\*\*\*

**description:** primary sequence record identifier.  
**example:**  
**default value:** n/a  
**ASN.1 struct:** For all db's except PDB and 'other':  
Bioseq->Seq-id->Textseq-id->accession  
For PDB sequences: this field is n/a  
For other databases: this field is n/a

**parser:** mother, update  
**function:** WriteACCDB  
**API:** SHoundAccFromGi returns this column's value given a gi.

\*\*\*chain\*\*\*

**description:** 'chain' describes which one of (possibly) many chains in a structure a sequence refers to. This only refers to PDB sequences.  
**example:** A  
**default value:** n/a  
**ASN.1 structure:** Bioseq->Seq-id->PDB-seq-id->chain  
**parser:** mother  
**function:** FillACCDB, given a Bioseq pointer, retrieves a Seq-id

**pointer.** If the Seq-id is of the type, pdb (protein data bank), then a PDB-seq-id ptr is passed to WriteACCDB which retrieves the value of chain.

**API:** SHoundGiFromPDBchain

**more info:** Sequences that are part of structures will have an associated PDB identifier. This identifier must be accompanied by info on the chain that this sequence refers to in the structure. A PDB identifier points to a structure with (possibly) many chains. A PDB identifier supplemented with a chain id will point to a single sequence (GI). For example, see PDB id "9XIM" and chain "A"; this points to GI 443580. See example below 9XIM\_A

Chain in the ASN.1 annotation is of type character not of type string so it appears in its decimal value. So 65 is A and so. NCBI's GenPept for 443580 lists chain 65 in DBSOURCE but if you look at structural display (in NCBI or PDB) the chains are always denoted A,B

\*\*\*release\*\*\*

**description:** release date of the database

**example:** 07-OCT-2004

**default value:** n/a

**ASN.1 structure:** For all db's except PDB, Giimport and 'other':  
Bioseq->Seq-id->Textseq-id->release (a string)  
For PDB sequences:  
Bioseq->Seq-id->PDB-seq-id->rel (a string or a NCBI data type)  
For Giimport:  
Bioseq->Seq-id->Giimport-id->release (a string)  
For other databases:  
this field is n/a

**parser:** mother

**function:** FillACCDB, given a Bioseq pointer, retrieves a Seq-id pointer. If the Seq-id is of the type, pdb (protein data bank), then a PDB-seq-id ptr is passed to WriteACCDB which retrieves the value of rel. If the Seq-id is of the type, genbank, embl, pir, swissprot, other, ddbj, prf, tpg, tpd or tpe then a Textseq-id pointer is passed to WriteACCDB which retrieves the value of release. If the Seq-id is of the type, giimport then a Giimport-id ptr is passed to WriteACCDB which retrieves the value of release. In all cases, the formatting of the date is attempted with NCBI's function DatePrint (like this 07-OCT-2004). If the Formatting cannot be done, then the string is just copied.

**API:** SHoundGiFromPDBchain  
SHoundSeqIdFromGi

\*\*\*version\*\*\*

**description:** version of the database

**example:** 1

**default value:** 0

**ASN.1 structure:** Bioseq->Seq-id->Textseq-id->release (an INTEGER)

**parser:** mother

**function:** FillACCDB, given a Bioseq pointer, retrieves a Seq-id pointer. If the Seq-id is of the type, genbank, embl, pir, swissprot, other, ddbj, prf, tpg, tpd or tpe then a Textseq-id pointer is passed to WriteACCDB which retrieves the value of version. For all other databases, this value is n/a

**API** This value cannot be directly retrieved from the table by the API but a Seq-id can be retrieved using SHoundSeqIdFromGi given a gi.

\*\*\*title\*\*\*

**description:** this is a short description of the sequence record.

**example:** DNA-directed RNA polymerase II largest subunit (B220)

**default value:**

**ASN.1 structure:** Bioseq->Seq-descr->title or Bioseq->Seq-descr->PDB-Block->compound

if its not found in one of these two places then the `GetTitle` tries `GetName` which retrieves the protein long descriptive name from the annotation:

Bioseq->Seq-annot->Seq-feat->Prot-ref->name

if `GetName` returns NULL too then `GetTitle` looks for title in the Bioseq-set description:

Bioseq-set->Seq-descr->title or Bioseq-set->Seq-descr->PDB-Block->compound

**parser:** mother, update

**function:** `GetTitle` (which may call `GetName`)

**API:** `SHoundTitleFromGi`

You could use `SHoundGetDefLine` which gets title and all identifiers if `SHoundTitleFromGi` is not available in your language choice

**more info:** note that `GetName` does NOT fill the column called name in this table

**histdb table**

Last updated August 5, 2004

**Database:** seqhound

**Table:** histdb

**Module:** hist

**Definition:** parses accdb and creates a history table of the GIs and the date of addition to the histdb and the action taken on the gi.

MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
gi	int(11)	No	0	GenInfo Identifier from accdb
acc	char(20)	No		accession
version	int(11)	Yes	NULL	version
date	char(8)	No	0	current date of when histdb updated
action	int(11)	No	0	choice of ACTION_ADDED, ACTION_KILLED, or ACTION_CHANGED

MySQL Indexes

Keyname	Type	Field
ihistdb_rowid	INDEX	rowid
ihistdb_gi	INDEX	gi
ihistdb_date	INDEX	date
ihistdb_action	INDEX	action

**Observation:**

Histdb is populated once using the accdb, and then it gets updated on a nightly basis. During the initial population, *histparser.c* extracts gi-accession number pairs from the accdb database, along with the version number (see accdb). These fields get stored in histdb, along with the current date. During the nightly updates, new records are added to the core sets of databases (parti, accdb, nucprot, taxgi, pubseq, sengi, sendb). These update files are release from NCBI as asn1 and refseq files. For example: *nc0506.aso.gz* from *ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily-nc/* and *rsnc0506.2004.bna.gz* from *ftp://ftp.ncbi.nih.gov/refseq/daily/*. For each of these files, the file is first passed to *update.c* (*slri/seqhound/updates*) and then to *mother.c* (*slri/seqhound/parsers*). In *update.c*, the gi and accession numbers are extracted from the bioseq structures. Update then searches for the gi-accession number pairs in SeqHound. If neither the gi nor the accession are found, then this is a new record (ACTION\_ADDED). If the accession is found in SeqHound but is associated with a different gi, then the record represents a changed record (ACTION\_CHANGED), and if the gi-accession pair is present in SeqHound, then the record represents a deleted record (ACTION\_KILLED). The gi, accession, action along with the version and date (current date update executed) get logged in histdb. After histdb has been updated, then the core databases are updated accordingly. Gi's with ACTION\_ADDED: will get added to accdb, nucprot, taxgi, pubseq, sengi, sendb, parti by mother. This runs after the histdb are updated. Gi's with ACTION\_CHANGED: The old records in the core databases will be deleted and the new updated information will be added to the core databases when mother runs. Gi's with ACTION\_KILLED: The old record information in the core databases are now obsolete and will be deleted from the core databases and the new information will be added to the databases once mother runs. For any gi that is killed, there will be another gi (with the same accession) that is added.

**Source org:** NCBI  
**Source file:** accdb for the initial population  
\*.aso.gz from *ftp://ftp.ncbi.nih.gov/ncbi-asn1/daily-nc/rsnc.\*.2004.bna.gz* from *ftp://ftp.ncbi.nih.gov/refseq/daily/*  
where \* is the current date formatted as MMDD  
**parser:** histparser - fills the initial population, update - makes the daily updates

### \*\*\*gj\*\*\*

**description:** primary seqhound record identifier  
**example:** 21676275  
**default value:** n/a  
**ASN.1 struct:** Bioseq-->Seqid->id (choose 12)  
**parser:** histparser (for initial population from accdb) update (for subsequent daily updates from the ASN structure)  
**function:** Extract\_ACCDB (*histparser.c*) LogHistory & ToBioseqUp (*update.c*)  
**API:** n/a. The updated information is stored in the core databases. histdb is used only to log the changes and is not used by external users.

### \*\*\*acc\*\*\*

**description:** GenInfo identifier for a sequence record.  
**example:** BQ628626  
**default value:** n/a  
**ASN.1 struct** For all databases except PDB and 'other': Bioseq->Seq-id->Textseq-id-

>accession.

For PDB sequence: this field is n/a.

For other databases: this field is n/a

**parser:** histparser (extracts the accession from accdb) update (parses the accession from the ASN structures)

**function:** Extract\_ACCDB (histparser), LogHistory & ToBioseqUp  
(*update.c*)

**API:** SHoundGetAccFromRetiredGi

### \*\*\*version\*\*\*

**description:** version of the database

**example:** 1

**default value:** 0

**ASN.1 struct** For all databases except PDB and 'other': Bioseq->Seq-id->Textseq-id->version.

For PDB sequence: this field is n/a.

For other databases: this field is n/a

**parser:** histparser (extracts the version from accdb) update (extracts the version from the ASN structures)

**function:** Extract\_ACCDB (*histparser.c*) LogHistory & ToBioseqUp  
(*update.c*)

**API:** n/a. The updated information is stored in the core databases. histdb is used only to log the changes and is not used by external users.

### \*\*\*date\*\*\*

**description:** date record was entered into histdb



**example:** 20040128 (YYYYMMDD)  
**default value:** current date  
**defined in:** this field is obtained through a system command  
**parser:** histparser (uses the current date) update (uses the current date)  
**function:** Extract\_ACCDB (histparser) calls GetStringDate LogHistory & ToBioseqUp (*update.c*)  
**API:** n/a. The updated information is stored in the core databases. histdb is used only to log the changes and is not used by external users.

**\*\*\*action\*\*\***

**description:** the action taken on the record  
**example:** ACTION\_ADDED, ACTION\_KILLED, ACTION\_CHANGED  
**default value:** n/a  
**defined in:** these values are macros in *hist\_db.h* (*slri/seqhound/include*)  
**parser:** histparser, update  
**function:** Extract\_ACCDB (*histparser.c*), LogHistory & ToBioseqUp (*update.c*)  
**API:** n/a. The updated information is stored in the core databases. histdb is used only to log the changes and is not used by external users.

**pubseq table**

Last updated August 5, 2004

**Database:** seqhound

**Table:** pubseq

**Definition:** This database maps a geninfo identifier to the medline article(s) that first publishes the gi.

MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
gi	int(11)	No	0	GenInfo Identifier
muid	int(11)	No	0	Medline identifier
pmid	int(11)	No	0	

MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	gi pmid
ipubseq_rowid	INDEX	rowid
ipubseq_gi	INDEX	gi
ipubseq_muid	INDEX	muid
ipubseq_pmid	INDEX	pmid

**Observation:** none

**Source org:** NCBI

**Source file:** \*.aso.gz from *ftp://ftp.ncbi.nih.gov/ncbi-asn1/*

**Parser:** mother

**\*\*\*gi\*\*\***

**description:** the gene identifier

**example:** 25992759

**default value:** n/a

**ASN.1 struct:** Bioseq-->Seqid (choice 12)

**parser:** mother

**function:** ToBioseq

**API:** SHoundGiFromReferenceID  
SHoundGiFromReferenceList

**\*\*\*muid\*\*\***

**description:** the medline id(s) from which the gi was first discussed

**example:** 22369064

**default value:** n/a

**ASN.1 struct:** Bioseq-entry->Bioseq->Seqdesc->Pubdesc->Pub-equiv->Pub->muid  
(choice 4)

**parser:** mother

**function:** ToBioseq

**API:** ShoundMuidFromGi  
ShoundMuidFromGiList  
SHoundGetReferenceIDFromGi  
SHoundGetReferenceIDFromGiList

SHoundMuidFrom3D  
SHoundMuidFrom3DList

**taxgi table**

Last updated August 5, 2004

**Database:** seqhound

**Table:** taxgi

**Module:** core

**Definition:** taxgi stores each GI even if it is not associated with any taxonomy Id  
MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
gi	int(11)	No	0	GenInfo Identifier
taxid	int(11)	No	0	Taxonomy Identifier
kloodge	int(11)	Yes	NULL	
type	varchar(50)	Yes	NULL	

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	gi
itaxgi_rowid	INDEX	rowid
itaxgi_gi	INDEX	gi
itaxgi_taxid	INDEX	taxid
itaxgi_kl	INDEX	kloodge
itaxgi_type	INDEX	type

**Observation.:** This table associates a 'gi' with a 'taxid'. Mother loops through all SeqEntries in ToBioseq(). When it finds a bioseq (SeqEntry->choice ==

1 ), mother will fill taxgi.

**Organization:** NCBI; chrom

**Source db:** \*.aso; chromff

**\*\*\*gi\*\*\***

**description:** GenInfo sequence record ID from GenBank.

**example:** 2313082

**default value:** n/a

**ASN.1 struct:** Bioseq-->Seq-id (choice 12)

**parser:** mother

**function:** mother calls ToBioseq() -> FillTAXGI() -> WriteTAXGI().  
update calls main()  
comgen calls AssignKlodge()  
from db\_layer: AppendRecordTAXGI(),  
SearchTAXGI(),  
SearchTAXGIDNA(),  
SearchTAXGIRNA(),  
SearchTAXGIProt(),  
EditRecordTAXGIByGI(),  
DeleteGIinTAXGI()

**API:** SHoundDNAFromTaxID()  
SHoundDNAFromTaxIDList()  
SHoundProteinsFromTaxID()  
SHoundProteinsFromTaxIDList()  
SHoundTaxIDFromGi()  
SHoundTaxIDFromGiList()

```
SHoundProteinsFromChromosome()  
SHoundProteinsFromChromosomeList()  
SHoundDNAFromChromosome()  
SHoundDNAFromChromosomeList()
```

**\*\*\*taxid\*\*\***

**description:** Taxonomy Identifier.

**example:** 3847

**default value:** n/a

**ASN.1 struct:** if Bioseq-->Seq-desc-Org-ref->Dbtag.db == "taxon"  
Bioseq-->Seq-desc-Org-ref->Dbtag.tag

**parser:** mother

**function:** mother calls ToBioseq() -> FillTAXGI() -> WriteTAXGI().  
comgen calls AssignKloodge()  
from db\_layer: AppendRecordTAXGI(),  
SearchTAXGI(),  
EditRecordTAXGIByGI(),  
DeleteGIinTAXGI()

**API:** SHoundTaxIDFromGi()  
SHoundTaxIDFromGiList()

**\*\*\*type\*\*\***

**description:** Type of Macro-molecule.

**example:** DNA  
RNA  
protein

NA  
other  
"not specified".

**default value:** n/a

**ASN.1 struct:** Bioseq -> Seq-desc -> GIBB-mil ->  
-> ENUMERATED  
{  
dna (0),  
rna (1),  
extrachrom (2),  
plasmid (3),  
mitochondrial (4),  
chloroplast (5),  
kinetoplast (6),  
cyanelle (7),  
synthetic (8),  
recombinant (9),  
partial (10),  
complete (11),  
mutagen (12), -- subject of mutagenesis ?  
natmut (13), -- natural mutant ?  
transposon (14),  
insertion-seq (15),  
no-left (16),-- missing left end (5' for na, NH2 for aa)  
no-right (17), -- missing right end (3' or COOH)  
macronuclear (18),  
proviral (19),



```
est (20), -- expressed sequence tag
sts (21), -- sequence tagged site
survey (22), -- one pass survey sequence
chromoplast (23),
genemap (24), -- is a genetic map
restmap (25), -- is an ordered restriction map
physmap (26), -- is a physical map
other (255)
}
```

**parser:**

mother

**function:**

mother calls `ToBioseq() -> FillTAXGI() -> WriteTAXGI()`.

update calls `main()`

comgen calls `AssignKloodge()`

from `db_layer`:

`AppendRecordTAXGI()`,

`SearchTAXGI()`,

`EditRecordTAXGIByGI()`,

`DeleteGIinTAXGI()`

**API:**

`SHoundDNAFromChromosome()`

`SHoundDNAFromChromosomeList()`

`SHoundProteinsFromChromosome()`

`SHoundProteinsFromChromosomeList()`

**\*\*\*kloodge\*\*\*****description:**

Kloodge is a running count of the number of completely sequenced chromosomes, if the gi listed in column 2 belongs to one of these complete chromosome sequence sets, then the kloodge 'id' is listed here.

**example:** 23

**default value:** n/a

**ASN.1 struct:** n/a

**parser:** comgen

**function:** `main() -> SLRIGetGIs() -> AssignKloodge()  
from db_layer:  
AppendRecordTAXGI(),  
SearchTAXGI(),  
SearchTAXGIDNA(),  
SearchTAXGIProt(),  
EditRecordTAXGIByGI(),  
DeleteGIinTAXGI()`

**API:**

**sengi table**

Last updated August 5, 2004

**Database:** seqhound

**Table:** sengi

**Module:** core

**Definition:** Seqentry to 'gi' conversion. Index table that stores partition names for sendb.

MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
gi	int(11)	No	0	GenInfo Identifier
seid	int(11)	No	0	GenInfo Identifier
division	char(15)	Yes	NULL	File Name of the partition.

MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	gi
isengi_rowid	INDEX	rowid
isengi_gi	INDEX	gi
isengi_seid	INDEX	seid

**Observation:** Seqhound keeps track of which Bioseqs come from which Bioseq-sets using the sengi table. In the sengi table, each Bioseq is represented by a GI (a unique identifier for the sequence record) and each Bioseq-set is represented by the first GI in the set of sequences.

The following paragraph is listed for historical purposes and will be deprecated in a future release.

In Codebase, the sengi database also acts as an index of sendb database files. Due to memory constraints in CodeBase (CodeBase cannot have file size > 2.0 GB), it is not possible to have one file to store all the ASN information. Instead, the information that should be stored in one sendb file is partitioned into divisions (based on the filename from which mother parses the information) and stored into numerous files that collectively represent the sendb database. For example, suppose that a request is made for a particular ASN structure given a gi. That gi is looked up in the parti database to determine which partition file the ASN information is stored in. Once the partition is obtained, the ASN.1 information is then extracted from that file.

**\*\*\*gi\*\*\***

**description:** GenInfo sequence record ID from GenBank.  
**example:** 2313082  
**default value:** n/a  
**ASN.1 struct:** Over Bioseq-->Seqid (choice 12)  
**parser:** mother  
**function:** mother calls ToBioseqSEQENTRY ( )  
from db\_layer:  
AppendRecordSENGI ( ), SearchSENGI ( ),  
DeleteGIinSENGI ( )

**API:** n/a

**\*\*\*seid\*\*\***

**description:** Seqentry Identifier.  
The 'gi' in the 1st column of this table describes a sequence that is part of a set of sequences (a seqentry from Genbank that is a Bioseqset). The 'gi' of the first sequence in this set of sequences will become the seqentry identifier (seid) that applies to all of the gi's in this set of seq

**example:** 231308

**default value:** n/a

**ASN.1 struct:** Over Seq-entry -> Bioseq-set -> Seq-entry -> Bioseq ->-> Seqid -> (choice 12) gi.

**parser:** mother

**function:** mother call recursive ToBioseqSEQENTRY( )  
db\_layer: AppendRecordSENGI( ) , SearchSENGI( ) .

**API:** n/a

**\*\*\*division\*\*\***

**description:** This is the name of sendb partition where the 'gi' listed in columns 1 can be found, generated from the NCBI file name parsed by mother.

**example:** gbest1675 or gbinv35 or gbmamls.

**default value:** n/a

**ASN.1 struct:** n/a This division name comes from the name of the file where the sequence was found..

**parser:** mother

**function:** Mother receive this division name as a parameter. (Name of the file

**API:** name that is parsed) db\_layer: AppendRecordSENGI ( )  
n/a

**sendb table**

Last updated August 5, 2004

**Database:** seqhound  
**Table:** sendb  
**Module:** core  
**Definition:** SeqEntry Data Base  
MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
seid	int(11)	No	0	GenInfo Identifier
asn1	mediumblob	No		ASN1 Binary SeqEntry.

**MySQL Indexes**

Keyname	Type	Field
PRIMARY	PRIMARY	seid
isendb_rowid	INDEX	rowid
isendb_seid	INDEX	seid

**Observation:** Sequences are distributed by GenBank in packages. Each package is referred to as a seq-entry.  
A seq-entry may contain either a single sequence record (called a Bioseq) or a set of sequence records (called a Bioseq-set). If a Seq-entry contains a Bioseq-set then that Bioseq-set contains a list of Seq-entry packages. (yes, this data structure is recursive). Each of these Seq-entry packages contains a single sequence record (a Bioseq). There is annotation that is associated with single sequence records (Bioseqs). An example of annotation is a list of authors who are responsible for

submitting a sequence record. There is also annotation associated with sets of sequence records (Bioseq-sets). This type of annotation applies to all of the sequence records that are in the set. For example a set of authors may be responsible for all of the sequence records in the set.

Seqhound stores Bioseqs and their associated GenInfo id's in one central table (called asndb). Since each of these Bioseqs may have come from a Bioseq-set, Seqhound needs a way to store the Bioseq-set associated annotation (that applies to each of the Bioseqs in the set). To accomplish this, Seqhound takes the Bioseq-set, removes the Bioseqs that it contains, and stores the remainder of the Bioseq-set (annotation) in the sendb table. Seqhound keeps track of which Bioseqs come from which Bioseq-sets using the sengi table. In the sengi table, each Bioseq is represented by a GI (a unique identifier for the sequence record) and each Bioseq-set is represented by the first GI in the set of sequences.

**\*\*\*seid\*\*\***

**description:** Will be whatever the first gi in the Seq-Entry gi's group.

**example:** 2313082

**default value:** n/a

**ASN.1 struct:** Over Seq-entry -> Bioseq-set -> Seq-entry -> Bioseq ->-> Seqid -> (choice 12) gi.

**parser:** mother

**function:** `main( )` in mother  
db\_layer:  
`OpenSENDB`  
`AppendRecordSENDB( )`



```
SearchSENDB( )  
DeleteGIinSENDB( )  
CloseSENDB( )
```

**API:** n/a

### \*\*\*asn1\*\*\*

**description:** ASN1 binary format containing a Bioseq-set).

**example:** n/a

**default value:** n/a

**ASN.1 struct:** Seq-entry (Bioseq-set)

**parser:** mother

**function:** main( ) in mother  
db\_layer:  
AppendRecordSENDB( )  
SearchSENDB( )  
DeleteGIinSENDB( )

**API:** This functions returns the Seqentry stored in this field after it first replaces all of the Bioseqs that belong in it.

```
SHoundGetBioseqSet( )  
SHoundGetSeqEntry( )
```

**Redundant protein sequences (redundb) module**

Last updated: August 4, 2004

**redund parser****purpose:**

The redund parser builds the redundb module, which consists of the redund table. The input file consists of information pertaining to redundant GIs, accession numbers and the sequence information which the GIs refer to. The resulting data table contain information on the GIs, redundant GI groups and the ranking of each GI within their redundant group.

**module:**

redundb

**input files:**

*nr.gz* from <ftp://ftp.ncbi.nih.gov/blast/db/FASTA/>

**table created:**

redund

**source code location:**

*slri/seqhound/parsers/redund.c*

**config file dependencies:**

*slri/seqhound/config/.intrezrc* (UNIX platform)

The relevant section of the configuration file is:

```
[datab]
;seqhound database that you are connecting
username=your user name
password=your pass word
dsn=dsn_in_odbc.ini_file
database=seqhound
local=
```

```
[sections]
;this should be set to 1 to allow usage of the redundb

;redundant protein sequences
redundb = 1
```

Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.) In section [sections] redundb should be 1.

### commandline parameters

Typing ***./redund*** at the command line will return a list of command line parameters and default settings. For example:

```
> ./redund
```

```
redund  arguments:
```

```
-i  Input non-redundant database fasta file [File In]
-n  Initialize database file [T/F]  Optional
    default = F
```

example use:

```
./redund -i nr -n T
```

### associated scripts:

*nrftp.pl*: retrieves the relevant data file from NCBI's ftp site

### error & run-time logs:

redund writes to a log file called *redundlog*

### additional information:

*ftp://ftp.ncbi.nih.gov/blast/db/FASTA/*

See data table descriptions for each of the tables that are listed under “tables altered”

**redund table**

Last updated: August 4, 2004

**Database:** seqhound  
**Table:** redund  
**Module:** redundb  
**Definition:** Partitions GIs into groups based on sequence redundancy, and ranks each GI within their group.

MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
gi	int(11)	No	0	GenInfo Identifier
rordinal	int(11)	No	0	Position in redundant group
rgroup	int(11)	Yes	NULL	ID of redundant group(not static)

MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	gi
iredund_rowid	INDEX	rowid
iredund_gi	INDEX	gi
iredund_ordinal	INDEX	rordinal
iredund_rgroup	INDEX	rgroup

**Observation:** redund is a database table that places GIs with equivalent sequences into redundant groups. The same sequence may be worked on by independent researchers. Each discovery is submitted and assigned a different gi. When the discoveries concern a common sequence, the GIs point to sequence records that describe redundant sequences, i.e. a single sequence has multiple GIs. These GIs then belong to a common group (rgroup). In such cases, it is common for one particular GI to be used more than the others because it is better annotated. The ordinal (rordinal) is a ranking of the GIs, 1 for the best annotated GI, 2 for the second, 3 for the third etc. This ordering is determined by the ordering of the sequence headers in the nr source file. These 3 pieces of information form the redund database table.

**Source org:** NCBI  
**Source file:** *nr.gz* from *ftp://ftp.ncbi.nih.gov/blast/db/FASTA/nr.gz*  
**parser:** redund

**\*\*\*gi\*\*\***

**description:** GenInfo Identifier for a sequence record.  
**example:** 7228451  
**default value:** 0  
**source:** parsed from nr (line 1 of record) as token(|) delimited fields  
**parser:** redund  
**function:** LabelGI, AssignRedund (*redund.c*)

**API:** SHoundRedundantGroup (returns all GIs in the same group as the input GI)  
SHoundRedundantGroupFromID (returns GIs from the group with input group ID)  
SHoundFirstOfRedundantGroupFromID (GI of rank 1, given a group)  
SHoundIsNRFirst (true/false, is this GI is ranked 1 in it's group)

**\*\*\*rordinal\*\*\***

**description:** rank position in redundant set  
**example:** 1  
**default value:** n/a  
**source:** based on the position of the gi within the record, the first gi has rordinal 1, the 2nd has ordinal 2  
**parser:** redund  
**function:** LabelGI, AssignRedund (*redund.c*)  
**API:** SHoundIsNRFirst (tells if a gi has rordinal = 1)

**\*\*\*rgroup\*\*\***

**description:** redundant group the gi is a member of  
**example:** 1  
**default value:** n/a  
**source:** rgroup IDs are assigned to GIs as the input file is parsed. Each GI encountered is assigned the rgroup ID of those GIs already encountered which have an identical sequence. If no such GIs have been encountered, the GI is assigned a new rgroup ID one number higher than the largest current rgroup ID in the database. The first GI encountered is given an rgroup ID of 1.  
**parser:** redund

**function:** LabelGI, AssignRedund (*redund.c*)  
**API:** SHoundRedundantGroupIDFromGI[List] (returns the ID of  
the rgroup that the input GI is a part of)

## Complete genomes tracking (gendb) module

### genff parsers:

#### purpose:

The genff parsers create the file '*genff*' that tracks complete genomes and the corresponding organism data (kloodge, taxid, type of chromosome, organism name, accession number of chromosome).

#### module

n/a

#### tables altered

*genff* file

#### source code location

*slri/seqhound/genomes/pregen.pl*

*slri/seqhound/genomes/gen.pl*

*slri/seqhound/genomes/ncbi.bacteria.pl*

*slri/seqhound/genomes/gen\_cxx.cpp*

#### config file dependency

n/a

#### command line parameters

n/a

#### logic of parsers

*genff* is built using 2 main scripts, *gen.pl* and *ncbi.bacteria.pl*. Both these scripts call *pregen.pl* and *gen\_cxx.cpp* and pass the necessary command line arguments.

*gen.pl* will create a new *genff* from scratch (backing up an existing *genff*) and populate it with eukaryotic complete genome data. This is done in two steps, first *pregen.pl* will pre-parse the asn files, and then *gen\_cxx.cpp* will parse the preparer-generated files and extract the relevant data into *genff*.

After *gen.pl* finishes executing, execute *ncbi.bacteria.pl* to add complete genome data from the bacteria data.

After your initial build, you can periodically update *genff* by running *ncbi.bacteria.pl* which will add only new bacteria complete genome data.

Both *gen.pl* and *ncbi.bacteria.pl* will check to make sure *pregen.pl* and *gen\_cxx* exists in the directory. *pregen.pl* is used to slim down the asn files. Without *pregen.pl*, building *genff* would take hours and require over 4 GB of RAM on your server. With *pregen.pl* the total run time is approximately 10 minutes and uses only a few MB of RAM.

#### additional info:

See [section 4.5](#) step 21



**chrom parser****purpose:**

The chrom parser creates the chrom data table that maps taxonomy id to their completely sequence genome information (Klodge, chromflags, access, name of chromosome).

**module:**

gendb

**input files:**

*slri/seqhound/genomes/genff*

**tables altered:**

chrom

**source code location:**

*slri/seqhound/genomes/chrom.c*

**config file dependencies:**

The relevant configuration file is:

*slri/seqhound/config/.intrezrc* (UNIX platform)

The relevant section is:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_.odbc.ini_file
database=seqhound
local=

[sections]
;complete genomes tracking
gendb = 1
```

Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.) In section [sections] gendb should be 1.

**command line parameters:**

Typing **./chrom** at the command line will return a list of command line parameters and default settings. For example:

**> ./chrom**

chrom arguments:

```
-i Input file [File In]
-n Initialize Input Database [T/F] Optional
```

```
default = F
```

**example use:**

```
./chrom -i ../genomes/genff -n T
```

**associated scripts:**

*slri/seqhound/scripts/ncbi.bacteria.pl*: compares the content of genff to the complete genome directory structure in NCBI's ftp site (*ftp://ftp.ncbi.nih.gov/genomes*) to determine newly sequenced genomes

(newly sequenced genomes have a directory in the ftp site, but no corresponding entry in genff). *ncbi.bacterial.pl* will print out a comparison of genff with the ftp site. Genff will have to be updated by manually adding newly sequenced genomes.

**error & run-time logs**

*chromlog*

**additional info:**

See data table descriptions for each of the tables that are listed under “tables altered”

**comgen parser****purpose:**

The comgen parser marks all proteins and DNA belonging to new completely sequenced Bacterial genomes (human and mouse completed) in SeqHound. It marks the complete genomes in taxgi by assigning a kloodge id to the taxgi table. Comgen also adds loci names into accdb if they are not already present.

The input files to comgen are ASCII ASN.1 formatted.

**module:**

gendb

**input files:**

*ftp://ftp.ncbi.nih.gov/genomes/\*/\*.asn*

**tables altered:**

taxgi, accdb

**source code location:**

*slri/seqhound/genomes/comgen.c*

**config file dependencies:**

n/a

**command line parameter:**

3 optional command line arguments.

```
-l Local data processing - all files
  default = T
-a Process one file with accession
  default = NULL
-p Path to data files
  default = ./
```

**example use:**

after running *slri/seqhound/scripts/genftp.pl* (which downloads the data files into a directory eg *./datafiles*):

```
./comgen -l T -p ./datafiles
```

**associated scripts:**

*genftp.pl* will download all the *\*.asn* files from

*ftp://ftp.ncbi.nih.gov/genomes/\**. The *\*.asn* files will then be used by comgen to update the data tables.

**error & run-time logs:**

*comgenlog*

**additional information:**

README's in *ftp://ftp.ncbi.nih.gov/genomes/*\*

See data table descriptions for each of the tables listed under “Tables altered”

**chrom table****Database:** seqhound**Table:** chrom**Module:** gendb**Definition:** This table maps taxonomy id with their complete genome information (Kloodge, chromflag, access, name)

## MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
taxid	int(11)	No	0	taxonomy identifier
kloodge	int(11)	No	0	Kloodge identifier
chromflag	int(11)	No	0	flag stating which type of chromosome (chromosome, plasmid, mitochondria, etc)
access	varchar(20)	No		accession for unit
name	text	No		unit name

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	kloodge
ichrom_rowid	INDEX	rowid
ichrom_taxid	INDEX	taxid
ichrom_kl	INDEX	kloodge
ichrom_acc	INDEX	access

**Observation:** This database keeps track of all completely sequenced chromosomes of a particular organism (**taxid**). The running count of completely sequenced units are tracked (**kloodge**); when another unit from any species is sequenced, it is added to chrom database and assigned the next incremented kloodge value. The type of sequenced unit is also stored (**chromflag**), the name (**name**) and accession number (**access**) of the sequenced unit is also stored. eg. For example, suppose at present, 190 chromosomes have been completely sequenced, then the highest kloodge value is 190. Assume also that Chromosome 4 (circular plasmid, named chr\_4, access Xchr\_4) of species X (taxid 123) is completely sequenced. Then a record is added to chrom database with fields:

```

taxid 123
kloodge 191
chromflag plasmid
accession Xchr_4

```

**Source org:** name chr\_4  
NCBI & Hogue Lab, Samuel Lunenfeld Research Institute (SLRI)  
**Source file:** slri/seqhound/genomes/genff  
**Parser:** chrom

**\*\*\*taxid\*\*\***

**description:** the NCBI taxonomy identifier  
**example:** 7226  
**default value:** 0  
**source:** genff, column 1  
**parser:** chrom  
**function:** AppendRecordCHROM  
**API:** SHoundAllGenomes (*intrez\_db.h*)

**\*\*\*kloodge\*\*\***

**description:** a running count of the number of completely sequenced chromosomes. Each completely sequenced chromosome is assigned a number 1 and upwards.  
**example:** 190  
**default value:** n/a  
**source:** genff, column 2  
**parser:** chrom  
**function:** AppendRecordCHROM  
**API:** SHoundAllKloodge (*intrez\_db.h*)  
SHoundAllKloodgeFromAccess (*intrez\_db.h*)

**\*\*\*chromflag\*\*\***

**description:** the type of chromosome (chromosome, plasmid, mitochondria etc) defined as byte macros in *intrez.h*.  
CHROM\_PHAGE (phage sequence)  
CHROM\_NR  
CHROM\_ECE  
CHROM\_PLMD (plasmid sequence)  
CHROM\_CHLO (chloroplast sequence)  
CHROM\_MITO (mitochondrial sequence)  
CHROM\_CHROM (chromosome sequence)  
CHROM\_ALL

**example:** CHROM\_ALL  
**default value:** n/a  
**source:** parsed from genff as an integer and converted into a byte (column 3)  
**parser:** chrom  
**function:** AppendRecordCHROM  
**API:** SHoundChromosomeFromGenome[List] (*intrez.h*)

\*\*\*access\*\*\*

**description:** the accession identifier of the chromosome  
**example:** mm\_chr10  
**default value:** n/a  
**source:** genff, column 4  
**parser:** chrom  
**function:** AppendRecordCHROM  
**API:** SHoundAllGenAccess (*intrez\_db.h*)

\*\*\*name\*\*\*

**description:** the scientific name of the organism  
**example:** Mus musculus chromosome 8  
**default value:** n/a  
**source:** genff, column 5  
**parser:** chrom  
**function:** AppendRecordCHROM  
**API:** GetNameFromCHROM (*intrez\_db.c*)

**Taxonomy hierarchy (taxdb) module**

Last **updated** August 18, 2004

**importtaxdb parser****purpose:**

The importtaxdb parser builds the taxonomy module, which consist of the taxdb, gcodedb, divdb, del and merge table. The input file consists of taxonomic information such as the taxonomy nodes, names, division, genetic codes, deleted nodes and merged nodes. Table taxdb holds taxonomy ids and a binary file associated with each id. Table gcodedb holds genetic code ids and a binary file associated with each id. Table divdb holds division ids and a binary file associated with each id. Table del holds the taxonomy id of the deleted nodes. Table merge holds the taxonomy ids of nodes which has been merged and which is result of merging.

**module:**

taxdb

**input files:**

*taxdump.tar.gz* from <ftp://ftp.ncbi.nih.gov/pub/taxonomy/>

**tables created:**

taxdb

gcodedb

divdb

del

merge

**source code location:**

*slri/seqhound/taxon/importtaxdb.c*

**config file dependencies:**

*slri/seqhound/config/.intrezrc* (UNIX platform)

The relevant section of the configuration file is:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_odbc_ini_file
database=seqhound
local=
```

```
[sections]
;taxonomy hierarchy
taxdb = 1
```



Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.) In section [sections] taxdb should be 1.

### **commandline parameters**

There are no command line parameters. Typing ***./importtaxdb*** will run the program:  
example use:

```
./importtaxdb
```

### **associated scripts:**

*taxftp.pl*: retrieves the relevant data file from NCBI's ftp site

### **error & run-time logs:**

importtaxdb writes to a log file called *importtaxdb\_log.txt*

### **additional information:**

readme files at [\*ftp://ftp.ncbi.nih.gov/pub/taxonomy/\*](ftp://ftp.ncbi.nih.gov/pub/taxonomy/)

See data table descriptions for each of the tables that are listed under “tables altered”

**taxdb table****Database:** seqhound**Table:** taxdb**Module:** taxdb**Definition:** Maps a taxid to information about the taxid. The information about the taxid is stored as an ASN blob.

## MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
tid	int(11)	No	0	Taxonomy identifier
asn	mediumblob	Yes	NULL	Information about tid

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	tid
itax_rowid	INDEX	rowid
itax_tid	INDEX	tid

**Observation:** The entire taxonomy can be viewed as a hierarchical taxonomy tree. The tree expresses the relationships between the nodes within the tree. At the root of the tree is a generic node that provides no information. Descendents of the root node include superkingdoms such as Archea, Eubacteria, Eukaryota, Viroids, Viruses. etc. Each of these superkingdoms have corresponding taxonomy children. Also included in the taxonomy tree are artificial sequences and unclassified taxonomies such as the prions, unidentified agents, etc. taxdb maps the taxonomy identifiers to the relevant information describing the taxonomies. Organizing the taxonomy as a tree allows users to request complete lineages, ancestors, children and permits the user to browse the tree exploiting its tree structure.

taxdb contains information from both *nodes.dmp* and *names.dmp*

**Source org:** NCBI

**Source files:** *nodes.dmp* from <ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz>  
*names.dmp* from <ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz>

**parser:** importtaxdb**\*\*\*tid\*\*\*****description:** the unique taxonomy identifier.**example:** 9606 (Human)

**default value:** n/a  
**source:** *nodes.dmp*: column 1  
**parser:** importtaxdb  
**function:** Parser\_TaxDBNodeRecord  
**API:** SHoundGetTaxChildNodes  
 SHoundGetTaxChildNodesList  
 SHoundGetAllTaxProgeny  
 SHoundGetTaxParent  
 SHoundGetAllTaxAncestors

---

Note that although relationships (parent, child) are not stored in the database, it is a part of the asn field, and as such it makes it possible to retrieve parent, children, ancestors and lineages (see asn blob).

---

### \*\*\*asn\*\*\*

**description:** ASN blob containing information about the taxonomy.  
 Fields in the ASN blob include:

- 1) parent taxonomy of node
- 2) rank of node
- 3) embl code for node
- 4) division of node (see divdb)
- 5) inherited div flag (1 if this node inherits its division from its parent)
- 6) genetic code (see gcodedb)
- 7) inherited genetic code (1 if this nodes inherits its genetic code from parent)
- 8) mitochondrial genetic code
- 9) inherited mit. genetic code (1 if nodes inherits mitochondrial gencode from parent)
- 10) genbank hidden flag (1 if name is suppressed in GenBank entry lineage)
- 11) subtree root flag (1 if this subtree has no sequence data yet)
- 12) comments
- 13) name of taxid (from *names.dmp*)

Possible rankings of node (there may be more):  
 superkingdom, kingdom  
 genus, subgenus  
 subspecies, species  
 subfamily, family, superfamily  
 phylum, subphylum,  
 subtribe, tribe

varietas  
 infraorder, order, suborder  
 infraclass, class, subclass  
 no rank

**example:**

A textual representation of the blob for taxid 9606:

```
SLRI-taxon ::= {
  taxId 9606 ,
  parent-taxId 9605 ,
  children-taxId {
    63221 } ,
  names {
    {
      name "Homo sapiens" ,
      name-class scientific-name } ,
    {
      name "human" ,
      name-class other ,
      other-class "genbank common name" } ,
    {
      name "man" ,
      name-class common-name } } ,
  rank {
    rank species ,
    premod none ,
    postmod none } ,
  embl-code "HS" ,
  division 5 ,
  inherited-div TRUE ,
  gencode 1 ,
  inherited-gencode TRUE ,
  mito-gencode 2 ,
  inherited-mito-gencode TRUE ,
  genbank-hidden FALSE ,
  hidden-subtree-root FALSE }
```

**default value:**

n/a

**source:**

*nodes.dmp*: column 2 - 13 and *names.dmp*: column 2, 3

**parser:**

importtaxdb

**function:**

Parser\_TaxDBNodeRecord

**API:**

The ASN object is not directly accessible through the API. Instead, fields in the ASN object may be retrieved and be accessible from the API. eg.

SHoundGetTaxNameFromTaxIDByClass

SHoundGetTaxNameFromTaxID

SHoundGetTaxLineageFromTaxID

You can also directly retrieve the ASN SLRITaxon by:

```
#include <taxdb_db.h>
AsnIoPtr aip = NULL;
SLRITaxonPtr ptax = NULL;
SHoundInit(FALSE, "name");
ptax = DB_GetTaxRec(9606);
aip = AsnIoNex(ASNIO_TEXT_OUT, stdout, NULL, NULL, NULL);
SLRITaxonAsnWrite(ptax, aip, NULL);
AsnIoClose(aip);
```

This will produce the text in the example above.



**gcodedb table**

**database:** seqhound  
**table:** gcodedb  
**Module:** taxdb  
**definition:** Maps a genetic code identifier to the corresponding ASN taxonomy genecode record.

## MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
gcid	int(11)	No	0	genetic code
asn	mediumblob	Yes	NULL	ASN taxon-gencode record

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	gcid
igcode_rowid	INDEX	rowid
igcode_gcid	INDEX	gcid

**Observation:** gcodedb is part of a group of databases (taxdb, gcodedb, divdb) that can be used to form a taxonomy hierarchy tree. In the case of gcode, the records map a genetic code identifier to an ASN object that holds information about that genetic code. taxdb (see taxdb later) will specify the type of genetic material each taxonomy uses, eg if the taxonomy has a plasmid as a genetic code, mitochondria DNA, or standard chromosomal DNA etc. gcodedb houses information concerning each of the genetic code. Relevant information include the translation table (the mapping of nucleic acid codons to amino acids), & the start codon. Further information at: <http://www.ncbi.nlm.nih.gov/Taxonomy>

**Source org:** NCBI

**Source file:** *gencode.dmp* from <ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz>

**Parser:** importtaxdb

**\*\*\*gcid\*\*\***

**description:** The genetic code identifier. The identifiers may not be consecutive because historically codes may have merged or new ones were created. The code and their corresponding names:  
 0 Unspecified  
 1 Standard  
 2 Vertebrate Mitochondrial

3 Yeast Mitochondrial  
 4 Mold Mitochondrial; Protozoan Mitochondrial; Coelenterate Mitochondrial; Mycoplasma; Spiroplasma  
 5 Invertebrate Mitochondrial  
 6 Ciliate Nuclear; Dasycladacean Nuclear; Hexamita Nuclear  
 9 Echinoderm Mitochondrial; Flatworm Mitochondrial  
 10 Euplotid Nuclear  
 11 Bacterial and Plant Plastid  
 12 Alternative Yeast Nuclear  
 13 Ascidian Mitochondrial  
 14 Alternative Flatworm Mitochondrial  
 15 Blepharisma Macronuclear  
 16 Chlorophycean Mitochondrial  
 21 Trematode Mitochondrial  
 22 Scenedesmus obliquus mitochondrial  
 23 Thraustochytrium mitochondrial code

**example:** Because many of the taxonomy groups are bacterial, 11 is a common genetic code.

**default value:** n/a

**source:** *gencode.dmp*: column 1

**parser:** importtaxdb

**function:** Parse\_TaxDBGenCodeRecord

**API:** n/a

\*\*\*asn\*\*\*

**description:** An ASN blob for the genetic code information corresponding to the genetic code identifiers. The individual fields in the blob are:

- 1) an optional abbreviation
- 2) name of genetic code (see gcid for names)
- 3) translation table
- 4) start codon

The translation table maps the nucleic acid codon to amino acids:

In the source file, the map is represented as:

`FLLSSSSYY**CC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGGG`

There are 64 characters in this map. Each position in the map corresponds to a codon (see below) and the letter at that position corresponds to the resulting amino acid that will be translated from that codon. The same can map be better expressed as a direct mapping from codon to amino acid where

- 1) first 3 letters represent a nucleic acid sequence (codon),
- 2) an amino acid letter identifier

## 3) an amino acid abbreviation

TTT	F Phe	TCT	S Ser	TAT	Y Tyr	TGT	C Cys
TTC	F Phe	TCC	S Ser	TAC	Y Tyr	TGC	C Cys
TTA	L Leu	TCA	S Ser	TAA	* Ter	TGA	* Ter
TTG	L Leu	TCG	S Ser	TAG	* Ter	TGG	W Trp
CTT	L Leu	CCT	P Pro	CAT	H His	CGT	R Arg
CTC	L Leu	CCC	P Pro	CAC	H His	CGC	R Arg
CTA	L Leu	CCA	P Pro	CAA	Q Gln	CGA	R Arg
CTG	L Leu	CCG	P Pro	CAG	Q Gln	CGG	R Arg
ATT	I Ile	ACT	T Thr	AAT	N Asn	AGT	S Ser
ATC	I Ile	ACC	T Thr	AAC	N Asn	AGC	S Ser
ATA	I Ile	ACA	T Thr	AAA	K Lys	AGA	R Arg
ATG	M Met	ACG	T Thr	AAG	K Lys	AGG	R Arg
GTT	V Val	GCT	A Ala	GAT	D Asp	GGT	G Gly
GTC	V Val	GCC	A Ala	GAC	D Asp	GGC	G Gly
GTA	V Val	GCA	A Ala	GAA	E Glu	GGA	G Gly
GTG	V Val	GCG	A Ala	GAG	E Glu	GGG	G Gly

The start codon is the particular codon that signals the start of protein translation for that particular genetic molecular (i.e. a standard chromosome starts translation at a different codon than a protozoan mitochondrial DNA).

The start codon is shown in the same manner as the translation table:

-----MM-----

This indicates that M (methionine) is the start codon.

Further information can be found at:

<http://www.ncbi.nlm.nih.gov/Taxonomy> in Genetic codes link.

**example:**

A textually representation of a blob for genetic code 11:

```
SLRI-taxon-gencode ::= {
    gencode-id 11 ,
    name "Bacterial and Plant Plastid" ,
    trans-table
    "FFLLSSSSYY**CC*WLLLLPPPPHHQQRRRRIIIMTTTTNNKKSSRRVVVVAAAADDEEGGG
    G " ,
    start-codons "---M-----M-----MMMM-----
    -M----- " }
```

**default value:**

n/a

**source:**

*gencode.dmp*: column 2, 3, 4, 5 are stored in the database as an ASN SLRITaxonGencode structure.

**parser**

importtaxdb

**function:**

Parse\_TaxDBGenCodeRecord

**API:**

not directly accessible from the API, but you can retrieve the SLRITaxonGencode object using:

```
#include <taxdb_db.h>
```

```
AsnIoPtr aip = NULL;
SLRITaxGencodePtr ptax = NULL;
SHoundInit(FALSE, "name");
```



```
ptax = DB_GetTaxGencodeRec(11);  
aip = AsnIoNex(ASNIO_TEXT_OUT, stdout, NULL, NULL, NULL);  
SLRITaxGencodeAsnWrite(ptax, aip, NULL);  
AsnIoClose(aip);
```

This will produce the text in the example above.

**divdb table**

**Database:** seqhound  
**Table:** divdb  
**Module:** taxdb  
**Definition:** Maps taxonomy division ID to an ASN SLRI-taxon-div object  
MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
did	int(11)	No	0	Taxonomy Division ID
asn	mediumblob	Yes	NULL	ASN taxonomy division record

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	did
idiv_rowid	INDEX	rowid
idiv_did	INDEX	did

**Observation:** divdb is part of a group of databases (taxdb, gcodedb, divdb) that can be used to form a taxonomy hierarchy tree. In the case of divdb, the records map a division identifier to an ASN object that holds information about that division. This information include a 3-character GenBank division code, a division name and an optional comment. There are 10 divisions (see below).

**Source org:** NCBI

**Source file:** *division.dmp* from *ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz*

**Parser:** importtaxdb

**\*\*\*did\*\*\***

**description:** The division identifier (0-11, see asn for the names of division).

**example:** 0

**source:** *division.dmp*: column 1

**parser:** importtaxdb

**function:** Parse\_TaxDBDivRecord

**API:** n/a

**\*\*\*asn\*\*\***

<b>description:</b>	<p>The ASN blob for the taxonomy division. The individual fields in the ASN blob are:</p> <ol style="list-style-type: none"> <li>1) a 3 character division code (see below)</li> <li>2) a division name (see below)</li> <li>3) an optional comment</li> </ol> <p>Possible division codes and corresponding division names are:</p> <p>BCT Bacteria  INV Invertebrates  MAM Mammals  PHG Phages  PLN Plants  PRI Primates  ROD Rodents  SYN Synthetic  UNA Unassigned  VRL Viruses  VRT Vertebrates</p>
<b>example:</b>	<p>the entire field is one ASN blob. A possible blob could be represented textually as:</p> <pre>SLRI-taxon-div ::= {     div-id 1 ,     div-code "INV" ,     div-name "Invertebrates" }</pre>
<b>source:</b>	Column 2, 3, and 4 of <i>division.dmp</i> are stored as one ASN blob (SLRITaxonDiv) created in-house.
<b>parser:</b>	importtaxdb
<b>function:</b>	Parse_TaxDBDivRecord (importtaxdb)
<b>API:</b>	<p>there is no API function that allows you to retrieve the entire ASN blob. The information is used to construct information about a taxid, (see taxdb). You can retrieve a TaxonDivRec using:</p> <pre>#include &lt;taxdb_db.h&gt;  AsnIoPtr aip = NULL; SLRITaxonDivPtr ptax = NULL; SHoundInit(FALSE, "name"); ptax = DB_GetTaxDivRec(1); aip = AsnIoNex(ASNIO_TEXT_OUT, stdout, NULL, NULL, NULL); SLRITaxonDivAsnWrite(ptax, aip, NULL); AsnIoClose(aip);</pre> <p>This will produce the text in the example above.</p>

**del table****Database:** seqhound**Table:** del**Module:** taxdb**Definition:** Contains all the deleted taxonomy identifiers.

MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
tid	int(11)	No	0	Taxonomy ID

MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	tid
idel_rowid	INDEX	rowid
idel_tid	INDEX	tid

**Observation:****Source org:** NCBI**Source files:** *delnodes.dmp* from *ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz***parser:** importtaxdb**\*\*\*tid\*\*\*****description:** The taxonomy identifier**example:** 15**default value:** n/a**source:** column 1 of *delnodes.dmp***parser:** importtaxdb**function:** main of importtaxdb**API:** SHoundIsTaxDeleted (true if taxid parameter is deleted)

**merge table****Database:** seqhound**Table:** merge**Module:** taxdb

**Definition:** Contains all tax nodes that have merged. Maps an old taxid before merging with its new taxid after the merge. A taxonomy may be deleted because initially it was considered species X, but when its DNA is analyzed, it is deemed to be insignificantly different from species Y, in which case one of the species gets merged.

## MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
otid	int(11)	No	0	old taxonomy id
ntid	int(11)	No	0	new taxonomy id

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	otid
imerge_rowid	INDEX	rowid
imerge_ntid	INDEX	ntid
imerge_otid	INDEX	otid

**Observation:****Source org:** NCBI**Source files:** *merged.dmp* from *ftp://ftp.ncbi.nih.gov/pub/taxonomy/taxdump.tar.gz***parser:** importtaxdb**\*\*\*otid\*\*\*****description:** the old taxid**example:** 12**default value:** n/a**source:** *merged.dmp*: column 1**parser:** importtaxdb**function:** main of importtaxdb**API:** SHoundIsTaxMerged (takes an old taxid and checks if it has been merged into a new taxid)

**\*\*\*ntid\*\*\*****description:** the new taxid**example:** 74109**default value:** n/a**source:** *merged.dmp*: column 2**parser:** importtaxdb**function:** main of importtaxdb**API:** SHoundIsTaxMerged (takes an old taxid and checks if it's been merged into a new taxid)

**Structural databases (strucdb) module**

Last updated October 6, 2004

**cbmmdb parser****purpose:**

The cbmmdb parser builds the strucdb module databases. The input files to cbmmdb contain data for experimentally determined macromolecular 3D structures. The data files are in ASN.1 format available from the NCBI ftp site.

**Module:**

Build the Strucdb module

Change to the mmdbdata directory.

```
cd $SEQH/7.mmdb.files
```

Create tables of the Strucdb module in the database.

Make sure file *strucdb.sql* has line `use seqhound` close to the beginning of the file.

```
mysql -u my_id -p -P my_port -h my_server < strucdb.sql
```

Where *my\_id*, *my\_port* and *my\_server* should be replaced by your userid for the database, the port of the database and the IP address or the server name of the database server respectively. You will be prompted to enter your password.

This creates tables mmdb, mmgi and domdb in the database.

strucdb

input files:

```
*.val.gz (ftp://ftp.ncbi.nih.gov/mmdb/mmdbdata)
```

**tables altered:**

mmdb, mmgi

**source code location:**

```
slri/seqhound/parsers/cbmmdb.c
```

**config file & other dependencies:**

The relevant configuration files are:

```
slri/seqhound/config/intrezrc (UNIX platform)
```

```
slri/seqhound/config/mmdbrc (UNIX platform)
```

```
ncbi/config/ncbirc (UNIX platform)
```

```
ncbi/data/bstdt.val (optional)
```

\* The requirement that *bstdt.val* be in the same directory as the parser is not strictly enforced. As long as *ncbirc* (see below) is properly configured, *bstdt.val* will be located. However, if *bstdt.val* is not in the same directory as parser, a warning will be issued in

cbmmdb's log file. Despite the warning, the parser should work properly, granted the configurations files are all set properly.

The *.mmdbrc* file should have at least 1 section [MMDB] with 3 fields:

- 1) Database: the path where the data source files (*\*.val.gz*) are located
- 2) Index: the index file for the data source files (*mmdb.idx*: this is also downloaded by *mmdbftp.pl*)
- 3) Gunzip: path to gunzip



sample *.mmdbrc* file:

```
[MMDB]
;Where the data source files are located
Database = ./
;Index of all data source files
Index = mmdb.idx
;Path to gunzip
Gunzip = /bin/gunzip
```

The *.ncbirc* file should have 1 section [NCBI] with 1 field:

1) Data: the path to *bstdt.val*

sample *.ncbirc* file:

In file *.ncbirc*, variable DATA should have a value which is the path of directory *ncbi/data* on your machine. File *.ncbirc* looks like (change text in italics):

```
[NCBI]
ROOT=/
DATA=/my_home/compile/ncbi/data/
```

```
[NCBI]
;where bstdt.val is located
Data=/ncbi/data
```

The *.intrezrc* file should have the pathmm set to where you want the databases to be stored.

sample *.intrezrc* file:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_odbc_ini_file
database=seqhound
local=

[sections]
;structural databases
strucdb = 1
```

Text in **italics** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.) In section [sections] strucdb should be 1.

*mmdb.idx* should be stored in the same directory as the data sources (\*.val.gz files). It is an index, containing a list of MMDB ID and PDB code pairs. The first line is the number

of Biostrucs in MMDB in the current release. The \* in \*.*val.gz* is the MMDB ID. The cbmmdb parser will initially parse the content of *mmdb.idx* to ensure its integrity.

**command line parameters:**

Typing "`./cbmmdb`" at the command line will return a list of parameters and default settings. For example:

```
> ./cbmmdb
```

```
cbmmdb  arguments:
```

```
  -n  Initialize Database File for Biostrucs [T/F]
  -m  Initialize Database File for MMDB Id and GI pairs
[T/F]
```

example use:

```
./cbmmdb -i T -m T
```

associated scripts:

*mmdbftp.pl* retrieves all the data files for input to cbmmdb.

error and run-time logs:

cbmmdb writes to *cbmmdblog*

**additional info:**

See additional readme files on the NCBI ftp site (*ftp://ftp.ncbi.nih.gov/mmdb*)

See data table descriptions for each of the tables that are listed under "tables altered"

**vastblst parser****purpose:**

The vastblst parser constructs the 3D domain information that is extracted from the MMDB 3D structures. vastblst builds the domdb table to store this information.

**module:**

strucdb

**input files:**

mmdb table

**tables altered:**

domdb

**source code location:**

*slri/seqhound/domains/vastblst.c*

**config file dependencies:**

*slri/seqhound/config/intrezrc* (UNIX platform)

The relevant section in the configuration file is:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_odbc_ini_file
database=seqhound
local=

[sections]
;structural databases
strucdb = 1
```

Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.) In section [sections] strucdb should be 1.

**command line parameters:**

Typing **./vastblst** at the command line will return a list of the command line parameters and default setting. For example:

```
> ./vastblst
```

```
vastblst arguments:
```

```
-n Initialize Database File for Domains [T/F]
```

**example use**

```
./vastblst -n T
```

**associated scripts:**

n/a

**additional information:**

See data table description for each table listed under “tables altered”

**pdbrep parser****purpose:**

The pdbrep parser fills in the 'rep' field in the domdb table. 'rep' is the best representative domain for that blast set. The input files consist of non-redundant PDB chain set, BLAST rankings for the set and the representative for that chain set. pdbrep stores the representative into the domdb table.

**module:**

strucdb

**input files:**

*nrpdb.\** from *ftp://ftp.ncbi.nih.gov/mmdb/nrtable*

**tables altered:**

domdb

**source code location:**

*slri/seqhound/domains/pdbrep.c*

**config file dependencies:**

The relevant configuration file is:

*slri/seqhound/config/.intrezrc* (UNIX platform)

The relevant sections are:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_odbc.ini_file
database=seqhound
local=

[sections]
;structural databases
strucdb = 1
```

Text in **italics** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.) In section [sections] strucdb should be 1.

**command line parameters:**

Typing `./pdbrep` at the commandline will generate a listing of commandline parameters. For example:

```
> ./pdbrep
```

```
pdbrep arguments:
```

```
-i Input nrpdb table [File In]
```

**example use:**

```
./pdbrep -i nrpdb.010400
```

associated scripts:

n/a

**error & run-time log:**

pdbrep writes to *pdbreplog*

**additional information:**

README's at <ftp://ftp.ncbi.nih.gov/mmdb/nrtable>

see data table descriptions for each of the tables that are listed under "tables altered"

**mmdb table**

**Database:** seqhound  
**Table:** mmdb  
**Module:** strucdb  
**Definition:** Is a database of experimentally determined macromolecular 3D structures (Molecular Modeling DB).

## MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
mmdbid	int(11)	No	0	Molecular modeling database identifier
asn1	mediumblob	No		Biostruc blob
pdbid	varchar(20)	No		PDB id
bwhat	int(11)	Yes	NULL	types of molecules in the biostruc
models	int(11)	Yes	NULL	number of models in record
molecules	int(11)	Yes	NULL	number of molecules in record
size	int(11)	Yes	NULL	size of uncompressed biostruc
bzsize	int(11)	Yes	NULL	size of compressed biostruc

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	mmdbid
immdb_rowid	INDEX	rowid
immdb_mmdbid	INDEX	mmdbid
immdb_pdbid	INDEX	pdbid

**Observation:** Most 3D structures are obtained through X-ray crystallography and NMR-spectroscopy. MMDB is a subset of Protein Data Bank (PDB), excluding theoretical models. 3D structures can provide information pertaining to the biological function, mechanism of function, evolutionary history and relationships of a biomolecule.

**Source org:** NCBI  
**Source file:** \*.val.gz from *ftp://ftp.ncbi.nih.gov/mmdb/mmdbdata*  
**FTP script:** *slri/seqhound/scripts/mmdbftp.pl*  
**parser:** cbmmdb



**\*\*\*mmdbid\*\*\***

**description:** a unique, stable numerical identifier. MMDB ID are assigned when a new structure enters MMDB. If an entry is deleted from PDB, then its MMDB entry is obsolete. Obsolete structures get archived.

**example:** 10

**default value:** n/a

**ASN.1 struct:** Biostruc->Biostruc-id (choice 1)

**parser:** cbmmdb

**function:** Biostruc2Modelstruc (*ncbi/biostruc/mmdbapi1.c*)

**API:** SHound3DExists

**\*\*\*asn1\*\*\***

**description:** the Biostruct blob (see ASN structure below). Contains information about the 3D structure of a molecule including information about the researchers, molecule bond information, etc.

**example:** see below

**default value:** n/a

**source:** \*.val.gz

**parser:** cbmmdb

**function:** MMDBBiostrucGet

**API:** SHoundGet3DfromPdbId  
SHoundGetPDB3D  
SHoundGet3D

**\*\*pdbid****\*\*\***

**description:** a 4 character identifier from the Protein Data Bank.

**example:** 100D

**default:** n/a

**ASN.1 struct:** Biostruc->Biostruc-id(choice 2)

**parser:** cbmmdb

**function:** Biostruc2Modelstruc (*ncbi/biostruc/mmdbapi1.c*)

**API:** no functions return the pdbid from mmdb, it is used as a key to retrieve a 3D structure, see asn1 API.

**\*\*\*bwhat****\*\*\***

**description:** an integer that represents the type of molecule.  
The possible flags are (defined in *ncbi/biostruc/mmdbapi1.h*) and their corresponding byte values are:

- 1) AM\_ION 0x80
- 2) AM\_RNA 0x40
- 3) AM\_WAT 0x20
- 4) AM\_SOL 0x10
- 5) AM\_HET 0x08
- 6) AM\_DNA 0x04
- 7) AM\_PROT 0x0
- 8) AM\_POLY 0x01
- 9) AM\_UNK 0x00

**example:** AM\_ION  
**default:** AM\_UNK  
**ASN.1 stuct:** Biostruc->Biostruc-graph->Molecule-graph->Bio-mol-descr->molecule-type  
**parser:** cbmmdb  
**function:** Biostruc2Modelstruc (*ncbi/biostruc/mmdbapi1.c*)  
**API:** SHound3DbWhat

#### \*\*\*models\*\*\*

**description:** an enumeration of the different models for this structure  
**example:** 3  
**default:** n/a  
**source:** this is enumerated by NCBI code, increment for EACH Biostruc->Biostruc-model in the ASN structure  
**parser:** cbmmdb  
**function:** Biostruc2Modelstruc (*ncbi/biostruc/mmdbapi1.c*)  
**API:** n/a, no public interface provided to this field

#### \*\*\*molecules\*\*\*

**description:** an enumeration of the number of molecules in the structure  
**example:** 3  
**default:** n/a  
**source:** this is enumerated by NCBI code, increment for EACH Biostruc->Biostruc-graph->Molecule-graph in the ASN structure  
**parser:** cbmmdb  
**function:** Biostruc2Modelstruc (*ncbi/biostruc/mmdbapi1.c*)  
**API:** n/a, no public interface provided to this field

#### \*\*\*size\*\*\*

**description:** the size of the uncompressed biostruc

**example:** 7691  
**default:** n/a  
**source:** the size of the uncompressed biostruc is determined just before it is compressed and appended  
**parser:** cbmmdb  
**function:** AssignASNMemBZMemo (*intrez\_cb.c*)  
**API:** n/a, no public interface provided to this field

\*\*\*bysize\*\*\*

**description:** the size of the compressed biostruc  
**example:** 31  
**default:** n/a  
**source:** the compressed size is determined after it is compressed  
**parser:** cbmmdb  
**function:** AssignASNMemBZMemo (*intrez\_cb.c*)  
**API:** n/a, no public interface provided to this field

**mmgi table**

**Database:** seqhound  
**Table:** mmgi  
**Module:** strucdb  
**Definition:** A mapping of mmdbid to gi.  
MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
mmdbid	int(11)	No	0	mmdb identifier
gi	int(11)	No	0	GenInfo Identifier

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	mmdbid gi
immgi_rowid	INDEX	rowid
immgi_mmdbid	INDEX	mmdbid
immgi_gi	INDEX	gi

**Observation:**

**Source org:** NCBI  
**Source file:** \*.val.gz from *ftp://ftp.ncbi.nih.gov/mmdb/mmdbdata*  
**FTP script:** *slri/seqhound/scripts/mmdbftp.pl*  
**parser:** cbmmdb (see mmdb for initialization info)

## \*\*\*mmdbid\*\*\*

**description:** 3D molecular model unique identifier  
**example:** 1  
**ASN.1 struct:** Biostruc->Biostruc-id (choice 1)  
**parser:** cbmmdb  
**function:** Main  
**API:** SHound3DExists  
SHound3DFromGi  
SHound3DFromGiList

## \*\*\*gi\*\*\*

**description:** GenInfo Identifier

**example:** 1420979  
**ASN.1 struct:** Biostruc->Biostruc-id  
**parser:** cbmmdb  
**function:** Main  
**API:** SHoundGiFrom3D  
SHoundGiFrom3DList

**domdb table**

**Database:** seqhound  
**Table:** domdb  
**Module:** strucdb  
**Definition:** Stores the information of structural domains.  
MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto number row identifier
mmdbid	int(11)	No	0	mmdb identifier
asn1	mediumblob	No		asn blob
pdbid	varchar(20)	No		pdb identifier
pdbchain	varchar(10)	Yes	NULL	pdb chain
gi	int(11)	No	0	geninfo identifier
domno	int(11)	Yes	NULL	domain number
domall	int(11)	No	0	number of domains in the whole structure
domid	int(11)	No	0	vast domain id
rep	int(11)	Yes	NULL	representative of blast set

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	domid
idomdb_rowid	INDEX	rowid
idomdb_mmdbid	INDEX	mmdbid
idomdb_pdbid	INDEX	pdbid
idomdb_gi	INDEX	gi
idomdb_domall	INDEX	domall
idomdb_domid	INDEX	domid

**Observation:** VAST (vector alignment search tool) is an NCBI-developed algorithm used to identify similar 3-D structures that may not possess sequence similarities. Structural information is taken from the mmdb database. 3D structures are converted into vectors (which corresponding to structurally significant regions of the protein). Adjacent vectors are grouped into domains. Domain information is used to determine distant homologs and provide structural information. The domains of proteins can be compared with other protein domains. Those with similar domain overlaps are likely to be distant homologs. The domdb database stores the domain information.

**Source org:** NCBI  
**Source file:** MMDB database & *nrpdb.\* from ftp://ftp.ncbi.nih.gov/mmdb/nrtable*  
**Parser:** vastblst & pdbrep  
**Note:** vastblst parser uses SHoundGet3D to retrieve the MMDB information..

### \*\*\*mmdbid\*\*\*

**description:** the 3D structure id from which the domain was computed.  
**example:** 3446  
**default value:** n/a  
**source:** retrieved from mmdbid by calling SHoundAllMMDBID.  
**parser:** vastblst  
**function:** Main calls SHoundAllMMDBID and MakeAModelstruc  
**API:** not used by the API, there are other tables with more complete mmdbid's.

### \*\*\*asn1\*\*\*

**description:** the asn structure of the domain. The structure gives the gi of the domain and the starting and ending point of the domain.

**example:** a text output of the SLRIVALNode:

```
SLRIVALNode ::= {
    domain {
        gi 443581 ,
        from 2 ,
        to 393 } }
```

**default value:** n/a

**source:** extracted from the biostruc reconstructed from what is stored in mmdb.

**parser:** vastblst

**function:** WriteFASTABByDomain calls MakeDomain

**API:** SHoundGetDomain to actually print out the ASN structure, after calling SHoundGetDomain, the return value (a SLRIVALNodePtr object) must be opened and streamed into an asn IO stream. eg.

```
{ // start of program
  SLRIVALNodePtr pdom = NULL;
  AsnIoPtr aip = NULL;

  pdom = SHoundGetDomain(443581, 0);
  aip = AsnIoNew(ASNIO_TEXT_OUT, stdout, NULL, NULL, NULL);
  SLRIVALNodeAsnWrite(pdom, aip, NULL);
  AsnIoClose(aip);
} // end of program, text output as in example
```

SHoundGetFastaDomain (to retrieve the text, follow code above,

replace SLRIVALNode with SLRIFasta

\*\*\*pdbid\*\*\*

**description:** the PDB id of the 3D structure from which the domain was computed  
**example:** 9XIM  
**default value:** n/a  
**ASN.1 struct:** n/a  
**parser:** vastblst  
**function:** WriteFASTABByDomain  
**API:** SHoundGetDomain

\*\*\*gi\*\*\*

**description:** the gi of the chain from which the domain was computed  
**example:** 443581  
**default value:** n/a  
**source:** extracted from the biostruc object reconstructed from data in mmdb.  
Biostruc->Biostruc-id (choice 1)  
**parser:** vastblst  
**function:** WriteFASTABByDomain  
**API:** SHoundGiFromPDBchain  
This field can also be parsed from the SLRIVALNode and SLRIFasta ASN structures (above).

\*\*\*domno\*\*\*

**description:** a protein chain may have several domains, each domain is enumerated, eg if a protein X has 10 domains, the first domain has domno 0, the second domain has domno 1, ...etc  
**example:** 0  
**default value:** n/a  
**source:** this is enumerated by NCBI code  
**parser:** vastblast  
**function:** WriteFASTABByDomain  
**API:** n/a, by itself, the domno provides little information, so it is not provided in the API

\*\*\*domall\*\*\*

**description:** The total number of domains in the 3D structure. In the above example protein X, domall is 10



**example:** 1  
**default value:** n/a  
**source:** this is enumerated by NCBI code  
**parser:** vastblast  
**function:** WriteFASTABByDomain  
**API:** n/a

\*\*\*domid\*\*\*

**description:** the id for the domain. domid is computed by hashing various id's, including (but not limited to) mmdbid, domno, etc.  
**example:** 34460200  
**default value:** n/a  
**source:** hashed in WriteFASTABByDomain  
**parser:** vastblast  
**function:** WriteFASTABByDomain  
**API:** n/a

\*\*\*rep\*\*\*

**description:** domains get grouped based on properties. The domains are then compared to the other domains in their group and then the best over domain representative of that group is chosen. This field reflects that representative.  
**example:** 1  
**default:** 0  
**source:** *nrpdb.\** from *ftp://ftp.ncbi.nih.gov/mmdb/nrtable* columns 6, 9 & A  
**parser:** pdbrep  
**function:** Main  
**API:** n/a

## Protein sequence neighbours (neighdb) module

Last updated: August 18, 2004

---

Note: The neighbours tables are precalculated on a cluster and the resulting tables are distributed in MySQL format on our ftp site. Therefore, this section is provided for informational purposes only, or for those who would like to build neighbours tables from their own sequence data; it is not necessary if one wishes simply to include the neighbours module into their own seqhound instance, in which case they should simply download the precomputed tables.

Note: the nblast development tree is in the slri directory on the same level as SeqHound (i.e.; *slri/nblast*).

---

This documentation includes a description of the nblast program.

The *NBLAST paper* is freely available from BioMed Central and is provided in the supplementary documents directory distributed with this manual. See:

Dumontier M, Hogue CW. *NBLAST: a cluster variant of BLAST for NxN comparisons*. **BMC Bioinformatics**. 2002 May 8;3(1):13. Epub 2002 May 08. PMID: 12019022

**Purpose:** This section describes how to build and use nblast, a program for generating a database of sequence neighbours.

### Installing nblast:

There are two ways to install nblast:

- \* Download nblast binaries for your platform from the sourceforge website:

[http://sourceforge.net/project/showfiles.php?group\\_id=17918](http://sourceforge.net/project/showfiles.php?group_id=17918)

or

- \* Compile **nblast** on your system: In order to compile nblast, two pieces of third party software are prerequisite: NCBI's C toolkit and Sequiter software's CodeBase database library. Nblast currently only supports CodeBase as its database backend, but it may be ported to other databases in the future. If your system meets these requirements, download the platform independent source code for nblast from the link above and proceed.

The NCBI C toolkit can be downloaded from NCBI's website at

[ftp://ftp.ncbi.nih.gov/toolbox/ncbi\\_tools/ncbi.tar.gz](ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools/ncbi.tar.gz)

CodeBase is a commercial database software library available from Sequiter software.

A description of how to compile and set up the nblast software follows. If you installed the binaries, or already have compiled the software, skip to the Configuration section. The installation procedure for CodeBase and the NCBI C toolkit are not detailed here, but are assumed to have been successful.

Once you have unpacked the source code tree, change directories into the topmost *nblast* directory and then exercise one of the following sets of instructions, depending on whether you are using a Unix or Windows system.

Unix:

- Modify the *nblast.mk* file for different source and library paths.
- Modify the *make.nblast* file or project settings to create an executable that incorporates NBLAST (-D NBLAST\_API), logging (-D NBLAST\_LOG), and/or MoBiDiCK(-D MOBIDICK\_API) (The MoBiDiCK library is currently not publicly available).
- From the *src* directory type '**make -f make.nblast**', if the *nblast* binary is built then build *nblastcleanup* with '**make -f make.nblastcleanup**'. The binaries will be placed in the *build/* subdirectory of the source tree.

Windows:

- Open the *nblast.dsw* workspace in MSVC.

#### Build NBLAST and NBLASTCLEANUP

The two *nblast* executables, *nblast* and *nblastcleanup*, should now be present in the *slri/nblast/build/* directory. Set your *PATH* environment variable appropriately so that you can execute them. Before you can use *nblast*, you'll have to do some more configuration, described in the next section.

#### Configuration of *nblast* environment:

At least one configuration file is required to use *nblast* and it's associated applications, more if you wish to have some of the relevant files outside of the current directory. At the very least, you must have an *nblast* configuration file, called *.nblastrc* on Unix, and *nblast.ini* on Windows. This file must be present either in your current directory or your home directory[may not apply on windows] and has the following formats:

*.nblastrc* on Unix:

```
; NBLAST configuration file
[NBLAST]
writepath = /home/nblast/build/
```

*nblast.ini* on Windows:

```
[NBLAST]
writepath = g:\code\slri\nblast\build
```

where the directory path named after "writepath =" specifies where the files generated by *nblast* will be written to. A trailing slash(unix) or backslash(windows) is necessary for the directory path to work.

If you want to place the scoring matrix (used by the blast algorithm) and/or the formatted fasta database in any directory but the current working one, you must also have an *ncbi* configuration file:

*.ncbirc* on Unix:

```
[NCBI]
```

```
DATA = g:\code\ncbi\data
```

```
[BLAST]
```

```
BLASTDB = g:\blastdb\
```

*ncbi.ini* on Windows:

```
[NCBI]
```

```
DATA = /code/ncbi/data
```

```
[BLAST]
```

```
BLASTDB = /blastdb/
```

Where the path following “DATA =” specifies the path to find the scoring matrix file, and the path following “BLASTDB” specifies the path to find the formatted fasta database.

The BLAST algorithm uses the BLOSUM62 scoring matrix. The file containing this matrix is required for nblast to function. The file is named *BLOSUM62*, and comes with the NCBI C toolkit. It should also come with the binary distribution of nblast.

Once the configuration files have been properly set, you are ready to run nblast, detailed in the next section.

## Running NBLAST

- \* Format the fasta database using formatdb: Before Nblast can process the protein sequences, they must be properly formatted. This is done using the formatdb program, available from NCBI, which takes a fasta formatted database of protein sequences and processes them into a form which can be BLASTed. Once you have downloaded formatdb, format your fasta database using the following command:

```
formatdb -oT -i <dbname>
```

Where **<dbname>** is the file name of your fasta formatted protein sequence database. Generally this is *nr*, the non-redundant protein sequence database, which can be downloaded from <ftp://ftp.ncbi.nlm.nih.gov/blast/db/nr.tar.gz> file (at the time of this writing, that file is ~500 MB compressed). **<dbname>** will be used throughout the rest of this document to mean the filename of your original fasta formatted sequence database.

- \* *Use NBLAST to build the skeleton database:* Before nblast can compare sequences, it needs to build a skeleton of the nblastdb table with information on how the sequences will be ordered and indexed. This is done by running nblast with the -N1 command line argument as follows:

```
nblast -i<dbname> -d<dbname> -N1
```

- \* *Use Nblast to do the blast comparisons.* Before the neighbour tables can be built, nblast must do the pairwise blast comparisons. In addition to the nblast specific command line options mentioned here, you may also apply options related to NCBI's

blastall to augment the BLAST results. For small databases, it is feasible to do this step on a single computer. Single computer execution:

```
nblast -i<dbname> -d<dbname> -e<eval_cutoff> -N2
```

Where **<eval\_cutoff>** is the maximum eval value allowed for BLAST comparisons between sequences. BLAST comparisons which result in values higher than this are not saved.

Parallel Computer Execution: Since these comparisons are computationally intensive for large databases (like nr), nblast is capable of splitting up the task across multiple compute nodes. This is done by distributing the **<dbname>N.\*** files generated by the last step, and the **<dbname>.p\*** files generated by formatdb, to the compute nodes, and then running nblast on each compute node with node specific options:

```
nblast -i<dbname> -d<dbname> -e<eval_cutoff> -N2 -C<node_#> -D<total_#of_nodes>
```

Where **<total\_#of\_nodes>** is the total number of compute nodes in your parallel computing system, and **<node\_#>** is the index (ranging from 1 to **<total\_#of\_nodes>**) of the current compute node with respect to the set of nodes which the nblast task is divided across. **<eval\_cutoff>** has the same meaning as in single computer usage.

Both single and parallel computer execution of the N2 mode of nblast results in creation of files entitled **<dbname>B<node\_#>.\*** (for single computers, **<node\_#>** defaults to 1). These contain the set of blast results which that particular compute node generated.

- \* *Build the nblastdb table of sequence neighbours:* Now the blastdb table of pairwise blast results can be used to build/fill the nblastdb table of neighbouring sequences. The pairwise BLAST results generated during the last step are processed and consolidated to generate the records for the neighbouring sequences table, nblastdb. Single Computer Execution:

```
nblast -i<dbname> -d<dbname> -e<eval_cutoff> -N4
```

Parallel Computer Execution: You must first collect the database files generated on the compute nodes during the last step onto the head node before running this step. Then run the following on the head node:

```
nblast -i<dbname> -d<dbname> -e<eval_cutoff> -N4 -C<node_#> -D<total_#of_nodes>
```

- \* *Cleanup the database and generate number of neighbours fields:* The neighbours table, nblastdb, is nearly complete. All that needs to be done is to fill in the #Neighbours fields of the nblastdb table. This is done using the program nblastcleanup with the following arguments:

```
nblastcleanup -bT -pT -aT -qT -d<dbname> -n<num_blastdbs>
```

Where **<num\_blastdbs>** is the number of blastdb's to check/build from, generally equal to the number of compute nodes you used (1 in the single computer case).

Your Neighbours database should now be completely built.

## NBLAST Update Procedure

Nblast was designed to process NCBI's non-redundant sequence database, named nr. This database contains a non-redundant list of protein sequences and their associated GenInfo identifier numbers (GIs). The NCBI's nr database is updated on a regular basis, with some GI's being removed, and others being added. It is desirable to keep the NeighDB module's nblastdb and blastdb tables updated with the nr database, without having to recompute all the neighbours. This is the purpose of nblast's update procedure, which removes GI's which have been "killed" from any entries in the nblast tables which contain them, and BLASTs and inserts "new" GI's where they are appropriate. This process involves multiple steps, which are outlined and explained here.

- \* Format new version of the fasta database using formatdb: Use formatdb to format your new version of the fasta database in the same way you did before. Note that your new fasta database should have the same name as the one you initially used to build the nblast tables.

**formatdb -oT -i <dbname>**

- \* *Run Nblast in Update mode N5:* This step compares the GIs in the out of date nblastdb with those in the updated FASTA database. It removes entries of the killed GIs from the neighbour lists of all gi's, deletes records which only exist because of the killed GI, and inserts empty records for the newly added GIs in the nblastdb. It also creates an update file called *NBLAST\_UPDATE.val* which lists which GI's have been killed and which are to be added, for the next stages to use. On the head compute node, execute:

**nblast -i<dbname> -N5**

- \* *Run Nblast in Update mode N6:* This step performs the pairwise BLAST comparisons between the newly added sequences and all the old sequences that haven't been killed, as well as comparing the new sequences with each other. Single Computer Execution:

**nblast -i<dbname> -e<eval\_cutoff> -N6**

Parallel Computer Execution:

**nblast -i<dbname> -e<eval\_cutoff> -N6 -C<node\_#> -D<total\_#of\_nodes>**

Where **<eval\_cutoff>** is the maximum allowed eval for comparison results, as described earlier.

- \* *Rebuild the Neighbouring Sequences Table:* Now rebuild the nblast database using the same command you used to build it previously. Single Computer Execution:

**nblast -i<dbname> -d<dbname> -e<eval\_cutoff> -N4**

Parallel Computer Execution: First collect all the **<dbname>B<node#>.\*** files from the compute nodes onto the head node. Then run the following command:

**nblast -i<dbname> -d<dbname> -e<eval\_cutoff> -N4 -C<node\_#> -D<total\_#of\_nodes>**

- \* *Cleanup:* The cleanup procedure is done in the same manner as for the initial build of the neighbour tables. Single Computer Execution:

```
nblastcleanup -bT -pT -aT -qT -d<dbname> -n  
<num_blastdbs>
```

Where **<num\_blastdbs>** is the number of blastdb's to check/build from, generally equal to the number of compute nodes you used (1 in the single computer case).

Your Neighbours database should now be properly updated.

**nbraccess program\***

---

Note: Not available at time of release.



**BLASTDB table**

Last updated: August 4, 2004

**Database:** SeqHound  
**Module:** (Sequence) Neighbours  
**Source:** NBLAST  
**Source File:** Derived from NxN comparison of sequences from the NR database from <ftp://ftp.ncbi.nih.gov/blast/db/FASTA/nr.gz>  
**Parser/Application:** NBLAST <http://www.sourceforge.net/projects/slrtools> (source & binaries)  
**Published Documentation:** BMC Bioinformatics <http://www.biomedcentral.com/1471-2105/3/13/>  
**Definition:** Pairwise sequence alignments from NBLAST computation

---

Note: The neighbours tables are precalculated on a cluster and the resulting tables are distributed in MySQL format on our ftp site. Therefore, this section is provided for informational purposes only, or for those who would like to build neighbours tables from their own sequence data; it is not necessary if one wishes simply to include the neighbours module into their own seqhound instance, in which case they should simply download the precomputed tables.

---

## Codebase

Column Name	Index	NULL	Data Type	Size	Column Definition
UID	P	N	INTEGER/Hash Key	14	Perfect Hash for 2 Ordinal Numbers
EVAL	N	N	FLOAT	E-Value	12.7 Alignment Eval
ASNSA	N	N	MEMO	ASN.1	Modified SeqAlign for pairwise sequence alignment

## MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	Yes	NULL	Auto incremented id
uid	bigint(20)	Yes	NULL	Perfect Hash for 2 Ordinal Numbers
eval	decimal(12,7)	Yes	NULL	12.7 Alignment Eval
asnsa	mediumblob	Yes	NULL	Modified SeqAlign for pairwise sequence alignment

## MySQL Indexes:

Keyname	Type	Field
iblastdb_rowid	INDEX	rowid
iblastdb_uid	INDEX	uid

**\*\*\* uid \*\*\***

**Description:** 64 bit Perfect Hash Key generated from 2 32 bit Ordinal Numbers in nblastdb

**example:**

**default value:** none

**source:** NBLAST - a cluster computer extension of BLAST to compute NxN sequence comparisons

**functions:** ShoundGetBlastResult: Given two GIs, checks if alignment exists, returns NBlast-result-set else NULL  
SHoundGetBlastSeqAlign : Given two GIs, calls SHoundGetBlastResult and formats the NBlast-result-set to a SeqAlign

**\*\*\* eval \*\*\***

**Description:** The BLAST e-value reported for this alignment

**example:** 0.0001

**default value:** none

**function:** Used by NBLAST to create the NBLASTDB table using a user-specified minimum eval

**\*\*\*asnsa\*\*\***

**Description:** NBlast-Result-Set (ASN.1 definition is described in *slri/nblast/asn/Nblastasn.asn*)

- Stores the useful parts of a seqalign from the BLAST computation including the alignment, bitscore, eval

**NBLASTDB table**

Last updated: August 4, 2004

**Database:** SeqHound  
**Module:** (Sequence) Neighbours  
**Source:** NBLAST  
**Source File:** Derived from NxN comparison of the NR database from  
<ftp://ftp.ncbi.nih.gov/blast/db/FASTA/nr.gz>  
**Parser/Application:** NBLAST <http://www.sourceforge.net/projects/slritools> (source & binaries)  
**Published Documentation:** BMC Bioinformatics <http://www.biomedcentral.com/1471-2105/3/13/>  
**Definition:** Sequence neighbour lists from NBLAST computation  
 Codebase

Column_Name	Index	NULL	Data_Type	Size	Column_Definition
ORD	P	N	INTEGER Ordinal Number	10	The ordinal number of this entry (auto-increment)
GI	I	N	INTEGER GenInfo Id	10	The GenInfo Identifier for the sequence for which there are neighbours in the list
NUM	I	N	INTEGER# Neighbours	10	Number of neighbours in ASN.1 structure
ASNNBR	N	N	MEMO	ASN.1	ASN.1 structure containing the list of GIs whose sequences neighbour that of this entries GI, and the corresponding evalues of the alignments.

## MySQL

Field	Type	Null	Default	Column_Definition
rowid	int(11)	No		Auto incremented id
ord	int(11)	No	0	The ordinal number of this entry
gi	int(11)	No	0	The GenInfo Identifier for the sequence for which there are neighbours in the list
num	int(11)	No	0	Number of neighbours in ASN.1 structure

asnnbr	mediumblob	No		ASN.1 structure containing the list of GIs whose sequences neighbour that of this entries GI, and the corresponding evalues of the alignments.
--------	------------	----	--	--

MySQL Indexes:

Keyname	Type	Field
inblastdb_rowid	INDEX	rowid
inblastdb_ord	INDEX	ord
inblastdb_gi	INDEX	gi
inblastdb_num	INDEX	num

### \*\*\*ord\*\*\*

**Description:** The ordinal value of the entry in the database (starts with 1). The ordinal values from two GIs in a pairwise alignment are perfectly and reversibly hashed into/from the BLASTDB UID value. This approach saves on space required to store two 32 bit integers.

**Example:** 1

**Default Value:** NA

**Function:**

### \*\*\*gi\*\*\*

**Description:** The GenInfo Identifier of the query sequence

**Example:** 2495000

**Default Value:** NA

**API Function:** SHoundNeighboursFromGiEx: Given GI, an evalute cutoff and a limit of 100 returned results, returns FLinkSet containing list of neighbour GIs and their alignment e-value

Other functions that use SeqHound Functionality (Redundant Groups & Taxonomy Protein Lists)

SHoundNeighboursFromGi: Calls SHoundNeighboursFromGiEx, and if the GI is not found, searches through the list of redundant GIs for the respective sequence to find an equivalent GI for which there is neighbour information, returning the ShoundNeighboursFromGiEx results for that GI.

SHoundNeighboursFromGiList: Calls SHoundNeighboursFromGi with each GI in a valnode list

SHoundNeighboursFromTaxID: Calls ShoundNeighboursFromGiList for each GI in a given taxonomy (SHoundProteinsFromOrganism)

SHoundNeighboursOfNeighbours: Fetches the neighbours of supplied GI and each of their neighbours (limit 100)

SHoundNeighboursOfNeighboursList: Fetches the neighbours of supplied GI list and each of their neighbours

SHoundGiAndNumNeighboursList: returns a list of all the GIs with more than 0 neighbours

SHoundNumNeighboursInDB: Fetches the number of neighbours in the nblastdb

### \*\*\*num\*\*\*

**Description:** Quick lookup value to find out how many neighbours are in the ASN.1 structure. Can also be used to sort the GIs with most/least neighbours

**Example:** range of integer values

**Default Value:** 0

**Function:**

### \*\*\*asnbr\*\*\*

**Description:** NBlast-GiAndEval-set (ASN.1 definition is described in *slri/nblast/asn/NBlastasn.asn*) stores the GI and it's list of sequence neighbours (as GI, evalute pairs)

**Functions:** see above for GI

## Locus link functional annotations (lldb) module

### llparser

Last updated: August 5, 2004

#### purpose:

The llparser parses the LocusLink data files to create and update a set of tables that correlate curated sequence data and descriptive information about genetic loci. It retrieves information on official nomenclature, aliases, sequence accessions, phenotypes, EC numbers, MIM numbers, UniGene clusters, homology and map locations.

**module:** lldb

#### input files:

LL\_tmpl

From *ftp://ftp.ncbi.nih.gov/refseq/LocusLink/*

#### tables altered:

ll\_cdd, ll\_go, ll\_llink, ll\_omim

#### source code location:

*slri/seqhound/locuslink/llparser.c*

#### config file dependencies:

The relevant configuration file is:

*slri/seqhound/parsers/.intrezrc* (for Unix platforms) or

The relevant section of the configuration file is:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_odbc_ini_file
database=seqhound
local=

[sections]
;locus link functional annotations
lldb = 1
```

Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.) In section [sections] lldb should be 1.

#### command line parameters:

This parser does not have any command line parameter, Just type “**./llparser**” and it will parse the LL\_tmpl file. This input file must be in the same directory as the llparser executable.

**associated scripts:**

see “*slri/seqhound/scripts/llftp.pl*”

This script retrieves the input file or this parser.

**error and run-time logs:**

llparser writes to a log file called “*llparserlog*” where it writes Time Stamp, Error #, goparser.c line # and cause of the problem.

e.g.

```
===== [ May 21, 2003  3:49 PM ] =====  
ERROR: [000.000] {llparser.c, line 81} Main: Cannot find LL_tmpl file.
```

**troubleshooting:****additional info:**

The LocusLink web page is at NCBI <http://www.ncbi.nlm.nih.gov/LocusLink/>

The LocusLink *README* file describes the input file for this parser  
<ftp://ftp.ncbi.nih.gov/refseq/LocusLink/>

See data table descriptions for each of the tables in the lldb module.

**addgoid parser**

Last updated: August 5, 2004

**purpose:**

**Correlates sequence record gi's with GO annotation identifiers.**

The Gene Ontology flat file parser adds information to the supplements ll\_go table. This information correlates sequence records GI's with GO annotation identifiers. This information is retrieved from the *gene\_association.compugen.Genbank* and *gene\_association.compugen.Swissprot* files from [www.geneontology.org](http://www.geneontology.org).

Other GO annotation data is available at this site but it is not currently incorporated into SeqHound. This is actively being worked on at the time of writing.

---

Note that the input files used here do not contain PubMed Identifiers.

---

This parser is dependent on the following tables, asndb, parti, accdb and nucprot. This parser is also dependent on SeqHound API.

For this reason, the mother parser must be run before using the addgoid parser.

**module:** lldb

**input files:**

*gene\_association.compugen.Genbank*

*gene\_association.compugen.Swissprot*

from <ftp://ftp.geneontology.org/pub/go/gene-associations/>

**tables altered:**

ll\_go

**source code location:**

*slri/seqhound/locuslink/addgoid.c*

**config file dependencies:**

The relevant configuration file is:

*slri/seqhound/parsers/intrezrc* (for Unix platforms) or

The relevant section of the configuration file is:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_odbc_ini_file
database=seqhound
local=

[sections]
;locus link functional annotations
lldb = 1
```



Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables username, password, dsn, database in section [datab] should have the same values as USER, PASSWORD, DSN and DATABASE respectively in your *.odbc.ini* (see Step 10 in section 4.4.) In section [sections] lldb should be 1.

#### **command line parameters:**

Typing “**./addgoid -**” at the command line while in the directory where addgoid resides will return a list of command line parameters and default settings. For example:

```
> ./addgoid -  
pdbrep arguments:  
-i Input file [File In]
```

#### **associated scripts:**

see */slri/seqhound/goftp.pl*

The script retrieves the files used as input by the addgoid parser. This script also retrieves the three input files required by the goparser (see godb module).

#### **error and run-time logs:**

addgoid parser writes to a log file called *addgoidlog* where it writes Time Stamp, Error #, goparser.c line # and cause of the problem. For example

```
===== [ May 21, 2003 4:14 PM ] =====  
NOTE: CoreLib [002.003] {ncbifile.c, line 624} FileOpen("gene_association.compugen.Swissprot", "r")  
failed
```

#### **troubleshooting:**

##### **additional info:**

The NCBI LocusLink web page is at

<http://www.ncbi.nlm.nih.gov/LocusLink/>.

The Gene Ontology Consortium documentation is at :<http://www.geneontology.org/>.

See data table descriptions for each of the tables that are listed under the lldb module.

**ll\_omim table**

Last updated: August 5, 2004

**Database** seqhound**Table** ll\_omim**Module** lldb

**Definition** OMIM Online Mendelian Inheritance in Man.  
 Online Mendelian Inheritance in Man (OMIMTM) is a continuously updated catalog of human genes and genetic disorders. OMIM focuses primarily on inherited, or heritable, genetic diseases. It is also considered to be a phenotypic companion to the human genome project. OMIM is based upon the text Mendelian Inheritance in Man, authored and edited by Dr. Victor A. McKusick and a team of science writers and editors at Johns Hopkins University and elsewhere.

## MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto incremented id
ll_id	int(11)	No	0	Locus Link Identifier
omim_id	int(11)	No	0	OMIM Identifier.

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	ll_id omim_id
illomim_rowid	INDEX	rowid
illomim_llid	INDEX	ll_id
illomim_omimid	INDEX	omim_id

**Observation:****Organization:** NCBI**Source db:** *ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL\_tmpl.gz***\*\*\* ll\_id \*\*\*****description:** Locus link record identifier.**example:** 1**default value:** 0

**source:** First field in each locus link record. Eg:  
 >>1  
 is the record with locus link id 1.

**parser:** llparser.c --> ll\_parser.c  
**function:** LL\_ParseFile() --> LL\_LineParser  
db\_layer: LL\_Append2OMIM\_DB()  
**API:** SHoundLLIDFromOMIM()

\*\*\* omim\_id \*\*\*

**description:** OMIM Online Mendelian  
**example:** 138670  
**default value:** n/a  
**source:** After tag 'OMIM: '  
**parser:** llparser.c --> ll\_parser.c  
**function:** LL\_LineParser() --> LL\_ParseOMIM()  
db\_layer: LL\_AppendRecord() --> LL\_Append2OMIM\_DB()  
**API:** SHoundOMIMFromLLID()  
SHoundOMIMFromGi()  
SHoundOMIMFromGiList()

**ll\_go table**

Last updated: August 5, 2004

**Database:** seqhound  
**Table:** ll\_go  
**Module:** lldb  
**Definition:** Associates a locus link identifier with the gene ontology (GO) annotation.

MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto incremented id
ll_id	int(11)	No	0	Locus Link Identifier
go_id	int(11)	Yes	NULL	Gene Ontology ID.
pmid	int(11)	Yes	NULL	Pub Med ID.
evidence	varchar(50)	Yes	NULL	Evidence Code.

MySQL Indexes

Keyname	Type	Field
illgo_rowid	INDEX	rowid
illgo_llid	INDEX	ll_id
illgo_goid	INDEX	go_id
illgo_pmid	INDEX	pmid

**Observation.:**

**Organization:** NCBI  
**Source file:** *ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL\_tmpl.gz*  
**Parser:** ll\_parser.c

**\*\*\* ll\_id \*\*\***

**description:** Locus link identifier  
**example:** 2  
**default value:** n/a  
**source:** see above  
**parser:** *ll\_parser.c -> ll\_parser.c*  
**function:** *LL\_LineParser() -> LL\_ParseNPUnit*  
**API:** *SHoundLLIDFromGOIDAndECode ()*

**\*\*\*go\_id\*\*\***

**description:** Gene Ontology Identifier.  
**example:** 5717  
**default value:** n/a  
**source:** after tag GO, 4th element  
 e.g.:  
 GO: cellular  
 component | extracellular | IDA | GO:0005574 | GOA | na

**parser:** *ll\_parser.c* or *ll\_parser.c*  
**function:** `LL_LineParser()` -> `LL_ParseGOUUnit`  
**API:** `SHoundGOIDFromLLID()`  
`SHoundGOIDFromGi()`  
`SHoundGOIDFromRedundantGi`  
`SHoundGOIDFromGiList`  
`SHoundGOIDFromRedundantGiList`

**\*\*\*pmid\*\*\***

**description:** PubMed Identification.  
**example:** 3458201  
**default value:** n/a  
**source:** after tag GO, 6th element  
 e.g.:  
 GO: cellular component | extracellular | IDA | GO:0005574 | GOA | 345201

**parser:** *llparser.c* -> *ll\_parser.c*  
**function:** `LL_LineParser()` -> `LL_ParseGOUUnit`  
**API:** `SHoundGOPMIDFromGiAndGOID()`

**\*\*\*evidence\*\*\***

**description:** Every GO annotation must indicate the type of evidence that supports it; these evidence codes correspond to broad categories of experimental or other support. <http://www.geneontology.org/GO.evidence.html>

**example:** IC: Inferred by Curator.  
 ND: No biological Data available.  
 TAS: Traceable Author Statement.  
 IEA: Inferred from Electronic Annotation.

**default value:** n/a  
**source:** After a 'GO:' line, it's the 3rd. field element  
 eg:  
 GO: cellular  
 component | extracellular | IDA | GO:0005574 | GOA | 34520

1  
**parser:** `llparser.c -> ll_parser.c`  
**function:** `LL_LineParser() -> LL_ParseGOUnt()`  
**API:** `SHoundGOECodeFromGiAndGOID()`

**ll\_link table**

Last updated: August 5, 2004

**Database:** seqhound**Table:** ll\_link**Module:** lldb**Definition:** Associates a locus link id with a sequence identifier  
MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto incremented id
ll_id	int(11)	No	0	Locus Link Identifier.
gi	int(11)	No	0	Gene Info Identifier
map	text	Yes	NULL	Gene location in chromosome.

## MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	ll_id gi
illll_rowid	INDEX	rowid
illll_llid	INDEX	ll_id
illll_gi	INDEX	gi

**Observation.:** Maps a locus link id with a gi and its location in the chromosome. Not all locus link ids will have a gi. A gi may be specified as an NP gi or a XP gi ( experimentally determined ). If an NP gi is available, it will be used. If an NP gi is not available, then the XP gi will be used. If neither the NP nor XP gi is available, then no gi will be used.

**Organization:** NCBI**Source file:** *ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL\_tmpl.gz*

\*\*\* ll\_id \*\*\*

**description:** Locus Link Identifier.**example:** 2**default value:** n/a**source:** start of record  
eg.  
>> 2**parser:** llparser.c --> ll\_parser.c

**function:** LL\_AppendLLID\_DB()  
LL\_ParseFile() -> LL\_LineParser()  
**API:** SHoundLLIDFromGi()  
SHoundLLIDFromGiList()  
  
\*\*\*gi\*\*\*  
**description:** Gene Info Identifier  
**example:** 21071030  
**default value:** n/a  
**source:** in the NP tag, 2nd element  
eg  
NP: NP\_570602|21071030  
**parser:** llparser.c --> ll\_parser.c  
**function:** LL\_ParseFile() --> LL\_LineParser() -->  
LL\_Append2LLID\_DB()  
**API:** SHoundGiFromLLID()  
SHoundGiFromLLIDList()  
  
\*\*\*map\*\*\*  
**description:** Location of the gene in the Chromosome  
**example:** 19q13.4  
**default value:** n/a  
**source:** in the MAP tag: 1st element  
MAP: 19q13.4|RefSeq|C|  
**parser:** llparser.c --> ll\_parser.c  
**function:** LL\_ParseFile() --> LL\_LineParser()  
LL\_Append2LLID\_DB()  
**API:** SHoundLocusFromGi()



**ll\_cdd table**

Last updated: August 5, 2004

**Database:** seqhound**Table:** ll\_cdd**Module:** lldb**Definition:** Conserved Domain Database.

Maps a locus link id to a Conserved Domain ID.

Proteins often contain several modules or domains, each with a distinct evolutionary origin and function. The CDD database may be used to identify the conserved domains present in a protein sequence.

Conserved Domains can be summarized with multiple local sequence alignments. Computational biologists have compiled collections of such alignments representing conserved domains, and LocusLink imports them from three major sources:

- \* Smart, the Simple Modular Architecture Research Tool
- \* Pfam (UK), Pfam-A seed alignments
- \* COG (Clusters of Orthologous Groups)

Smart and Pfam are public domain databases, which are offered in combination with HMM-based search engines and alignment visualization services.

COG is an NCBI-curated protein classification resource. Sequence alignments corresponding to COGs are created automatically from constituent sequences and have not been validated manually for import in CDD.

The default e-value cutoff of for data in this table is 0.01

## MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto incremented id
ll_id	int(11)	No	0	Locus Link Identifier
cdd_id	varchar(50)	Yes	NULL	CDD Source Database.
evaluate	decimal(20,10)	Yes	NULL	Descriptive match.

## MySQL Indexes

Keyname	Type	Field
illcdd_rowid	INDEX	rowid
illcdd_llid	INDEX	ll_id
illcdd_cddid	INDEX	cdd_id

**Observation.:** Domains can be thought of as distinct functional and/or structural units

of a protein. These two classifications coincide rather often, as a matter of fact, and what is found as an independently folding unit of a polypeptide chain also carries specific function. Domains are often identified as recurring (sequence or structure) units, which may exist in various contexts. e.g.: of the CDD Record.

CDD: pfam00207: Alpha-2-macroglobulin  
family|5952|2318|na|8.970050e+02

CDDID: pfam00207

Evalue: 8.970050e+02

**Organization:** NCBI

**Source db:** *ftp://ftp.ncbi.nih.gov/refseq/LocusLink/LL\_tmpl.gz*

\*\*\* ll\_id \*\*\*

**description:** Locus link identifier

**example:** 2

**default value:** n/a

**source:** start of record

eg.

>>2

**parser:** llparser.c --> ll\_parser.c

**function:** LL\_AppendLLID\_DB( )

LL\_ParseFile( ) -> LL\_LineParser( )

**API:** SHoundLLIDFromCDDID( )

SHoundLLIDFromGi( )

SHoundLLIDFromGiList( )

\*\*\*cdd\_id\*\*\*

**description:** Conserved Domain Database Identifier

**example:** pfam00207

**default value:** n/a

**source:** after CDD tag: 1st element

CDD: pfam00207: Alpha-2-macroglobulin  
family|5952|2318|na|8.970050e+02

**parser:** llparser.c --> ll\_parser.c

**function:** LL\_ParseNPUnit( )

LL\_ParseFile( ) --> LL\_LineParser( )

**API:** SHoundCDDIDFromGi( )

SHoundCDDIDFromGiList( )

SHoundCDDIDFromLLID( )

**\*\*\*evaluate\*\*\***

**description:** Describes match between CDD and the sequence  
**example:** 8.970050e+02  
**default value:** 0  
**source:** after CDD tag: last element  
CDD: pfam00207: Alpha-2-macroglobulin  
family|5952|2318|na|8.970050e+02  
  
**parser:** llparser.c --> ll\_parser.c  
**function:** LL\_ParseNPUnit()  
LL\_ParseFile() --> LL\_LineParser()  
**API:** SHoundCDDScoreFromGi()

**GENE module****parse\_gene\_files.pl parser**

Last updated September 27, 2004

**purpose:**

*parse\_gene\_files.pl* parses 4 files from the NCBI Gene database and populates *gene\_\** tables in SeqHound.

**logic:**

*parse\_gene\_files.pl* first drops the *gene\_\** tables and then recreates them. It then reads the files generated by *gene\_cron.pl* to populate the new tables. In the future, the tables will not be dropped upon update.

**module: gene****input files:**

*ftp://ftp.ncbi.nih.gov/gene/DATA*

*gene2refseqUniq*

*gene\_infoUniq*

*gene\_historyUniq*

*gene2pubmedUniq*

**tables altered:** *gene\_dbxref*, *gene\_genomicgi*, *gene\_history*, *gene\_info*, *gene\_productgi*, *gene\_pubmed*, *gene\_object*, *gene\_synonyms*.

source code location:

*/seqhound/gene/parse\_gene\_files.pl*

*/seqhound/scripts/gene\_cron.pl*

**config file dependencies:**

The relevant configuration file is:

*slri/seqhound/config/.intrezrc* (for Unix platforms)

The relevant section of the configuration file is:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_.odbc.ini_file
database=seqhound
local=
```

Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables *username*, *password*, *dsn*, *database* in section [datab] should have the same values as *USER*, *PASSWORD*, *DSN* and *DATABASE* respectively in your *.odbc.ini* (see Step 10 in section 4.4.) *.intrezrc* must reside in the directory where the parser is running and *.odbc.ini* should be in your home directory.

You should also have a copy of *shconfig.pm* in the directory where the parser is running to read *.intrezrc* and *.odbc.ini*.

**command line parameters:****example use:**

```
perl parse_gene_files.pl
```

**associated scripts:**

Four files are downloaded from *ftp://ftp.ncbi.nih.gov/gene/DATA* by *gene\_cron.pl*:

*gene2refseq.gz*

*gene\_info.gz*

*gene\_history.gz*

*gene2pubmed.gz*.

*gene\_cron.pl* unzips the files and makes sure that each file only contains unique records generating the following files:

*gene2refseqUniq*

*gene\_infoUniq*

*gene\_historyUniq*

*gene2pubmedUniq*

**error and run-time logs:**

*parse\_gene\_files.log*

troubleshooting:

**additional info:**

<http://www.ncbi.nlm.nih.gov/entrez/query/static/help/genehelp.html>

---

Note that the NCBI gene database is experimental. This means that the input files and the SeqHound tables may change.

---

**gene\_dbxref table**

Last updated September 28, 2004

**Database:** SeqHound**Table:** gene\_dbxref**Module:** GENE**Definition:** The table holds cross references to other databases for GeneIds and their associated gis.

## MySQL

Field	Type	Null	Default	Column Definition
id	Int(11)	No		Identifier for a cross reference.
geneinfoid	Int(11)	No		Id of the gene_info record that this cross reference refers to.
dbname	Varchar(255)	No		Database name
dbaccession	Varchar(30)	No		Accession.

## MySQL Indexes

Keyname	Type	Field
igenedbxrefs_id	INDEX	id
igenedbxrefs_info	INDEX	geneinfoid
igenedbxrefs_db_name	INDEX	dbname
igenedbxrefs_db_access	INDEX	dbaccess

**Observation:****Source org:** NCBI**Source file:** *gene\_info.gz***FTP script:** *gene\_cron.pl***Parser:** *parse\_gene\_files.pl***\*\*\*id\*\*\*****description:** Identifier for the cross reference. Mysql auto-increment table.**example:** 1**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl*

**function:****API:** not available yet**\*\*\*geneinfo\*\*\*****description:** Identifier for the geneinfo record where this cross reference was found.**example:** 53577**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:****API:** not available yet**\*\*\*dbname\*\*\*****description:** Name of the database that contains the second reference.**example:** SGD**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:****API:** not available yet**\*\*\*dbaccess\*\*\*****description:** Accession for the record that contains the name.**example:** S0000572**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:****API:**

**gene\_genomicgi table**

Last updated September 29, 2004

**Database:** SeqHound**Table:** gene\_genomicgi**Module:** GENE**Definition:** Table that contains the gis of genomic DNAs that contain genes. The start, stop location and orientation on the genomic DNA are also included.

## MySQL

Field	Type	Null	Default	Column Definition
id	int(11)	No		Identifier of a product gi. Mysql auto-increment column.
geneobjectid	int(11)	No		Identifier of the geneobject that this genomic gi refers to.
gi	int(11)	No		The genomic gi.
start	int(11)	Yes		The start location on the genomic DNA.
end	int(11)	Yes		The start location on the genomic DNA.
orientation	char(1)	Yes		The orientation on the genomic DNA.

## MySQL Indexes

Keyname	Type	Field
igenomic_id		id
igenomic_objectid		geneobjectid
igenomic_gi		gi

**Observation:****Source org:** NCBI**Source file:** *gene\_info.gz***FTP script:** *gene\_cron.pl***Parser:** *parse\_gene\_files.pl*



**\*\*\*id\*\*\***

**description:** Identifier of a genomic gi.

**example:** 1

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:** not available yet

**\*\*\*geneobjectid\*\*\***

**description:** The gene object that this gi refers to.

**example:**

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:** not available yet

**\*\*\*gi\*\*\***

**description:** The genomic gi.

**example:** 10954454

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:** not available yet

**\*\*\*start\*\*\***

**description:** The start position of the gene on the genomic DNA.

**example:** 348

**default value:**

**ASN.1 struct:**

**source:**

**parser:**

*parse\_gene\_files.pl*

**function:**

**API:**

not available yet

**\*\*\*end\*\*\***

**description:**

The end position of the gene on the genomic DNA.

**example:**

1190

**default value:**

**ASN.1 struct:**

**source:**

**parser:**

*parse\_gene\_files.pl*

**function:**

**API:**

not available yet

**\*\*\*orientation\*\*\***

**description:**

The orientation of the gene on the genomic DNA.

- meaning "minus" strand or
- + meaning "plus" strand

**example:**

-

**default value:**

**ASN.1 struct:**

**source:**

**parser:**

*parse\_gene\_files.pl*

**function:**

**API:**

not available yet

**gene\_history table**

Last updated September 29,2004

**Database:** SeqHound**Table:** gene\_history**Module:** GENE**Definition:** This table has information about geneids that are no longer current.

## MySQL

Field	Type	Null	Default	Column Definition
id	int(11)	No		Identifier for a history record. Mysql auto-increment column.
taxid	int(11)	Yes		NCBI taxonomy identifier.
currentgeneid	int(11)	Yes		The current NCBI Gene Id for this gene.
oldgeneid	int(11)	Yes		The discontinued Gene Id.
oldsymbol	varchar(20)	Yes		The symbol assigned to the discontinued Gene Id, if the discontinued record was not replaced with another.

## MySQL Indexes

Keyname	Type	Field
igenehistory_id	Index	id
current_gene_id	Index	currentgeneid
discont_gene_id	Index	oldgeneid
discont_symbol	Index	oldsymbol

**Observation:** Note that all fields except for id are optional.**Source org:** NCBI**Source file:** *gene\_history.gz***FTP script:** *gene\_cron.pl***Parser:** *parse\_gene\_files.pl*

**\*\*\*id\*\*\***

**description:** Identifier for the history record. Mysql auto-increment column.

**example:** 212815

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:** not available yet

**\*\*\*taxid\*\*\***

**description:** The NCBI taxonomy identifier for this gene..

**example:** 10116

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:** not available yet

**\*\*\*currentgeneid\*\***

**\***

**description:** The current NCBI gene id.

**example:** 29666

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:** not available yet

**\*\*\*oldgeneid\*\*\***

**description:** The discontinued NCBI gene id.

**example:**

**default value:**

**ASN.1 struct:**

**source:****parser:** *parse\_gene\_files.pl***function:****API:** not available yet**\*\*\*oldsymbol\*\*\*****description:** The symbol associated with the discontinued gene id, if no new record replaced the discontinued record.**example:** hlyB**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:****API:** not available yet

**gene\_info table**

Last updated September 29, 2004

**Database:** SeqHound  
**Table:** gene\_info  
**Module:** GENE  
**Definition:** Information about a gene.

## MySQL

Field	Type	Null	Default	Column Definition
id	int(11)	No		Identifier for this record.
geneobjectid	int(11)	No		geneobjectid association with this information.
symbol	varchar(255)	Yes		The default symbol for this gene.
locustag	varchar(255)	Yes		The LocusTag for this gene.
chromosome	varchar(32)	Yes		The chromosome where this gene is found.
maplocation	varchar(255)	Yes		The map location of this gene on the chromosome.
description	mediumtext	Yes		A description of this gene.

## MySQL Indexes

Keyname	Type	Field
igeneinfo_id	Index	id
object_id	Index	geneobjectid
symbol	Index	symbol
locus_tag	Index	locustag
chr	Index	chromosome

**Observation:**  
**Source org:** NCBI  
**Source file:** *gene\_info.gz*  
**FTP script:** *gene\_cron.pl*  
**Parser:** *parse\_gene\_files.pl*

**\*\*\*id\*\*\***

**description:** Identifier for this record.

**example:** 74058

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:**

**\*\*\*geneobjectid\*\***

**description:** The geneobjectid for the gene that this information refers to.

**example:** 73870

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:**

**\*\*\*symbol\*\*\***

**description:** The symbol for this gene.

**example:** 1A981

**default value:**

**ASN.1 struct:** gene->locus (NCBI says this is where the data comes from).

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:**

**\*\*\*locustag\*\*\***

**description:** The locus tag for this gene.

**example:** 1A981

**default value:**

**ASN.1 struct:** gene->locus-tag (NCBI says this is where the data comes from).

**source:****parser:** *parse\_gene\_files.pl***function:****API:****\*\*\*chromosome\*\*\*****description:** The chromosome where this gene is found.**example:** I**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:****API:****\*\*\*maplocation\*\*\*****description:** The map location of this gene on the chromosome.**example:** I;-19.13 cM (interpolated genetic position)**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:****API:****\*\*\*description\*\*\*****description:** A description of this gene.**example:** putative protein (69.3 kD) (1A981)**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:****API:**



**gene\_object table**

Last updated September 29,2004

**Database:** SeqHound

**Table:** gene\_object

**Module:** GENE

**Definition:** This table contains information on the status of gene records.

## MySQL

Field	Type	Null	Default	Column Definition
id	int(11)	No		Identifier for this gene.
geneid	int(11)	No		NCBI's Gene Id.
status	varchar(64))	Yes		Status of the RefSeq for this gene. May be provisional, INFERRED, MODEL, PREDICTED, Reviewed or VALIDATED

## MySQL Indexes

Keyname	Type	Field
igeneobject_id	Index	id
igeneobject_geneid	Index	geneid
igeneobject_status	Index	status

**Observation:**

**Source org:** NCBI

**Source file:** *gene2refseq.gz*

**FTP script:** *gene\_cron.pl*

**Parser:** *parse\_gene\_files.pl*

**\*\*\*id\*\*\*****description:** Identifier for this gene.**example:** 8**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:****API:** not available yet**\*\*\*geneid\*\*****description:** NCBI's Gene Id.**example:** 1246510**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:****API:** not available yet**\*\*\*status\*\*\*****description:** The status of the RefSeq record that contains this gene.

NULL

INFERRED

MODEL

PREDICTED

Provisional

Reviewed

VALIDATED

**example:** Provisional**default value:****ASN.1 struct:****source:****parser:** *parse\_gene\_files.pl***function:**

**API:** not available yet

**gene\_productgi table**

Last updated September 29, 2004

**Database:** SeqHound  
**Table:** gene\_productgi  
**Module:** GENE  
**Definition:** Stores the gis of the RNA and Proteins produced by a gene.

## MySQL

Field	Type	Null	Default	Column Definition
id	int(11)	No		Internal identifier for the product gi. Mysql auto-increment column.
geneobjectid	int(11)	No		The geneobjectid for the gene that this gi belongs to.
gi	int(11)	No		The gi.

## MySQL Indexes

Keyname	Type	Field
iproduct_id	Index	id
iproduct_objected	Index	geneobjectid
iproduct_gi	Index	gi

**Observation:**

**Source org:** NCBI  
**Source file:** *gene2refseq.gz*  
**FTP script:** *gene\_cron.pl*  
**Parser:** *parse\_gene\_files.pl*

**\*\*\*id\*\*\***

**description:** Internal identifier for this gi. Mysql auto-increment column.  
**example:** 9  
**default value:**  
**ASN.1 struct:**  
**source:**  
**parser:** *parse\_gene\_files.pl*  
**function:**  
**API:** not available yet

**\*\*\*geneobjectid\*\*\***

**description:** NCBI's identifier for the gene that this gi belongs to.

**example:** 10

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:** not available yet

**\*\*\*gi\*\*\***

**description:** The gi. May be a protein or an RNA gi.

**example:** 32455275

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:** not available yet

**gene\_pubmed table**

Last updated September 29, 2004

**Database:** SeqHound  
**Table:** gene\_pubmed  
**Module:** GENE  
**Definition:** This table stores publications that refer to genes.

## MySQL

Field	Type	Null	Default	Column Definition
id	int(11)	No		Internal identifier for this reference.
geneobjectid	int(11)	Yes		The geneobjectid that this reference is about.
pmid	int(11)	Yes		The pmid of the reference.

## MySQL Indexes

Keyname	Type	Field
ipubmed_id	Index	id
ipubmed_geneid	Index	geneobjectid
ipmid	Index	pmid

**Observation:**

**Source org:** NCBI  
**Source file:** *gene2pubmed.gz*  
**FTP script:** *gene\_cron.pl*  
**Parser:** *parse\_gene\_files.pl*

**\*\*\*id\*\*\***

**description:** Internal identifier of this reference or of this gene objectid-pmid pair  
**example:** 112394  
**default value:**  
**ASN.1 struct:**  
**source:**  
**source:**  
**parser:** *parse\_gene\_files.pl*  
**function:**

**API:** not available yet

**\*\*\*geneobjectid\*\*\***

**description:** The geneobjectid of the gene.

**example:** 173392

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:**

**\*\*\*pmid\*\*\***

**description:** The pmid of the reference.

**example:** 8889548

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:**

**gene\_synonyms table**

Last updated September 28, 2004

**Database:** SeqHound  
**Table:** gene\_synonyms table  
**Module:** GENE  
**Definition:** Stores synonyms for genes.

## MySQL

Field	Type	Null	Default	Column Definition
id	int(11)	No		Identifier of this synonym, Mysql auto-increment column
geneinfoid	int(11)	Yes		The geneinfoid for this synonym.
synonym	text	Yes		The synonym.

## MySQL Indexes

Keyname	Type	Field
igenesyn_id	Index	id
igeneinfoid	Index	geneinfoid
isynonym	Index	synonym

**Observation:**

**Source org:** NCBI  
**Source file:** *gene\_info.gz*  
**FTP script:** *gene\_cron.pl*  
**Parser:** *parse\_gene\_files.pl*

**\*\*\*id\***

**description:** Internal identifier for this synonym geneinfoid pair. Basically a rowid. Mysql auto-increment column.  
**example:** 11179  
**default value:**  
**ASN.1 struct:**  
**source:**  
**parser:** *parse\_gene\_files.pl*  
**function:**



**API:****\*\*\*geneinfoid**

**description:** The geneinfoid for the record where this synonym was found.

**example:** 27686

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:**

**\*\*\*synonym**

**description:** The synonym.

**example:** T1J24.19

**default value:**

**ASN.1 struct:**

**source:**

**parser:** *parse\_gene\_files.pl*

**function:**

**API:**

## Gene Ontology hierarchy (godb) module

### goparser

Last updated: August 5, 2004

#### purpose:

Go data files contain information from the Gene Ontology Consortium that describe controlled vocabularies for the description of the molecular function, biological process and cellular component of gene products.

The GO parser is responsible for generating the hierarchical gene ontology datafiles: *go\_name*, *go\_parent*, *go\_synonym* and *go\_reference*. This is accomplished by parsing the flat files *component.ontology*, *function.ontology* and *process.ontology*.

**module:** godb

#### input files:

*component.ontology*

*function.ontology*

*process.ontology*.

From <ftp://ftp.geneontology.org/pub/go/ontology/>

#### tables altered:

go\_name, go\_parent, GO\_REF, GO\_SYN

#### source code location:

*slri/seqhound/go/goparser.c*

#### config file dependencies:

The relevant configuration file is:

*slri/seqhound/parsers/.intrezrc* (for Unix platforms) or

The relevant section of the configuration file is:

```
[datab]
;seqhound database that you are connecting
username=your_user_name
password=your_pass_word
dsn=dsn_in_odbc.ini_file
database=seqhound
local=

[sections]
;gene ontology hierarchy
godb = 1
```

Text in ***italics*** must be changed for the *.intrezrc* file to function correctly with your SeqHound set-up. Variables *username*, *password*, *dsn*, *database* in section [datab] should have the same values as *USER*, *PASSWORD*, *DSN* and *DATABASE* respectively in your *.odbc.ini* (see Step 10 in section 4.4.) In section [sections] *godb* should be 1.

**command line parameters:**

This parser does not have any command line parameters. Typing “**./goparser**” will process the three Gene Ontology files.

These files (*component.ontology*, *function.ontology* and *process.ontology*) must be present in the same directory as the compiled goparser executable

**associated scripts:**

see “*/slri/seqhound/goftp.pl*”

This script retrieves the three input files required by the goparser. The script also retrieves the files used as input by the addgoid parser (see lldb module).

**error and run-time logs:**

goparser writes to a log file called “*goparserlog*” where it writes Time Stamp, Error #, goparser.c line # and cause of the problem.

e.g.

```
===== [ May 21, 2003  2:36 PM=====
ERROR: [000.000] {goparser.c, line 74} Main: Cannot find function.ontology file
in current directory.
```

**troubleshooting:****additional info:**

The Gene Ontology Consortium documentation is located at  
<http://www.geneontology.org/>

See data table descriptions for each of the tables that are listed under SeqHound's Data Dictionary.

**go\_parent table**

Last updated: August 5, 2004

**Database:** Seqhound**Table:** go\_parent**Module:** Godb**Definition:**

MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto number row identifier
go_id	int(11)	No	0	GeneOntology ID
parent_goid	int(11)	No	0	Parent Go Id..

MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	go_id parent_goid
igoparent_rowid	INDEX	rowid
igoparent_goid	INDEX	go_id
igoparent_pid	INDEX	parent_goid

**Observation.:** Function, process and component are represented as directed acyclic graphs (DAGs) or networks. The difference between a DAG and a hierarchy is that in the latter each child can only have one parent; a DAG allows a child to have more than one parent. A child term may be an "instance" of its parent term (isa relationship) or a component of its parent term (part-of relationship). A child term may have more than one parent term and may have a different class of relationship with its different parents.

**Organization:** Gene Ontology (<ftp://ftp.geneontology.org/pub/go/ontology>)

**Source db:** function.ontology (Molecular function)  
 process.ontology (Biological process)  
 component.ontology (Cellular component)

\*\*\*go\_id\*\*\*

**description:** Gene Ontology identifier Child.**example:** 0016172

%antifreeze activity ; GO:0016172 {Parent}

%ice nucleation inhibitor activity ; GO:0016173 {Child}

**default value:** n/a

**source:** related to the indentation in the file.

**parser:** goparser.c -> go\_parser.c

**function:**

```
GODB_ParseFile()->GO_LineParser()-
>GO_AppendRecord()
  db_layer: GO_AppendRecord()->GO_Append_Name()->
GO_AppendParent()
GO_GetParentOf()
GO_GetChildrenOf()
GO_GetAllChildren()
GO_GetAllAncestors()
```

**API:** SHoundGODBGetChildrenOf()

\*\*\*parent\_goid\*\*\*

**description:** Gene Ontology identifier Parent.

**example:** 0016173

```
%antifreeze activity ; GO:0016172 {Parent}
%ice nucleation inhibitor activity ; GO:0016173 {Child}
```

**default value:** n/a

**source:** Related to indentation in the file

**parser:** goparser.c -> go\_parser.c

**function:**

```
GODB_ParseFile()->GO_LineParser()-
>GO_AppendRecord()
db_layer: GO_AppendRecord()->GO_Append_Parent()
GO_GetParentOf()
GO_GetChildrenOf()
GO_GetAllChildren()
GO_GetAllAncestors()
```

**API:** SHoundGODBGetParentOf()

**go\_name table**

Last updated: August 5, 2004

**Database:** seqhound**Table:** go\_name**Module:** godb**Definition:**

MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto incremented id
go_id	int(11)	No	0	GeneOntology ID.
go_name	text	No		Go_id Identifier name.
go_db	int(11)	No	0	GO Database name.
go_level	int(11)	No	0	Hierarchy level of the GO ID.

MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	go_id go_name go_db go_level
igoname_rowid	INDEX	rowid
igoname_goid	INDEX	go_id

**Observation.:** This table contains a list of 'go\_id' and it's molecular function name as well the database where the go\_id was parsed from and the level in the Gene Ontology hierarchy.

**Organization:** GeneOntology (<ftp://ftp.geneontology.org/pub/go/ontology>)

**Source db:** function.ontology (Molecular function)  
process.ontology (Biological process)  
component.ontology (Cellular component)

\*\*\*go\_id\*\*\*

**description:** Gene Ontology identifier.

**example:** 15643

**default value:** n/a

**source:** See example Go File. The GO id follows 'GO:' tag.

**parser:** goparser.c -> go\_parser.c

**function:** GODB\_ParseFile() -> GO\_LineParser() -  
>GO\_AppendRecord()  
db\_layer: GO\_AppendRecord() ->GO\_Append\_Name()  
GO\_GetRecordByID()

**API:** SHoundGOIDFromRedundantGi  
SHoundGOIDFromRedundantGiList  
SHoundGOIDFromGi  
SHoundGOIDFromGiList

### \*\*\*go\_name\*\*\*

**description:** Gene Ontology identifier name.

**example:** anti-toxin;

**default value:** n/a

**source:** go\_name follows, '%' and proceeds ';GO:'.

**parser:** goparser.c -> go\_parser.c

**function:** GODB\_ParseFile() -> GO\_LineParser() -  
>GO\_AppendRecord()  
db\_layer: GO\_AppendRecord() ->GO\_Append\_Name()  
GO\_GetNameByID()

**API:** SHoundGODBGetNameByID

### \*\*\*go\_db\*\*\*

**description:** Gene Ontology Database file. This is an integer 1, 2 or 3

**example:** 1  
where:  
1 GO\_MOLFUNCTION  
2 GO\_BIOPROCESS  
3 GO\_CELLCOMPONEN

**default value:** n/a

**source:** NA

**parser:** goparser.c -> go\_parser.c

**function:** GODB\_ParseFile() -> GO\_LineParser() -  
>GO\_AppendRecord()  
db\_layer: GO\_AppendRecord() ->GO\_Append\_Name()  
GO\_GetClassification()

**API:** SHoundGODBGetClassification()

### \*\*\*go\_level\*\*\*

<b>description:</b>	Hierarchy level of the GO ID. <hr/> <div>This is not used because the same GO ID can appear at different levels making the process to determine its level irrelevant. This was intended to be used to indicate the distance of a given GO node from the root node.</div> <hr/>
<b>example:</b>	3
<b>default value:</b>	n/a
<b>source:</b>	Related to the indentation of <code>'%{function name}'</code>
<b>parser:</b>	<code>goparser.c -&gt; go_parser.c</code>
<b>function:</b>	<code>GODB_ParseFile() -&gt; GO_LineParser() -&gt; GO_AppendRecord()</code> <code>db_layer: GO_AppendRecord() -&gt; GO_Append_Name()</code> <code>GO_GetHierarchyLevel()</code>
<b>API:</b>	<code>SHoundGODBGetHierarchyLevel()</code> this function is deprecated.



**go\_reference table**

Last updated: August 5, 2004

**Database:** seqhound  
**Table:** go\_reference  
**Module:** godb  
**Definition:**  
 MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto incremented id
go_id	int(11)	No	0	GeneOntology ID
go_ref	text	No		GeneOntology Reference.
go_ref_db	varchar(50)	Yes	NULL	Reference Data Base Name

MySQL Indexes

Keyname	Type	Field
PRIMARY	PRIMARY	go_id go_ref
igoref_rowid	INDEX	rowid
igoref_go_ref_db	INDEX	go_ref_db

**Observation.:** This table stores the 'go\_id' with its database cross-reference. The cross-reference can be an external database identifier that points to something that is equivalent to a given GO term.

**Organization:** GeneOntology (<ftp://ftp.geneontology.org/pub/go/ontology>)

**Source db:** function.ontology (Molecular function)  
 process.ontology (Biological process)  
 component.ontology (Cellular component)

\*\*\*go\_id\*\*\*

**description:** Gene Ontology identifier.

**example:** 45174

**default value:** n/a

**source:** The go\_id comes after '%{function name} ; GO:xxxxxx ; '

**parser:** goparser.c -> go\_parser.c

**function:** GODB\_ParseFile() -> GO\_LineParser() -  
 >GO\_AppendRecord()  
 db\_layer: GO\_AppendRecord() -

**API:** >Go\_Append\_Reference( )  
GO\_GetRecordByReference( )  
SHoundGODBGetRecordByReference( )

\*\*\*go\_ref\*\*\*

**description:** Gene Ontology Synonym Description.

**example:** ISBN: 0198506732

**default value:** n/a

**source:** go\_ref comes after "GO:xxxxxx ; ISBN: 0198506732 ;"

**parser:** goparser.c -> go\_parser.c

**function:** GODB\_ParseFile( ) -> GO\_LineParser( ) -  
>GO\_AppendRecord( )  
db\_layer: GO\_AppendRecord( ) -  
>GO\_Append\_Reference( )  
GO\_GetRecordByReference( )

**API:** NA

\*\*\*go\_ref\_db\*\*\*

**description:** Gene Ontology Reference Database.

**example:** EC  
ISBN  
TC

**default value:** n/a

**source:** After '%{function\_name} ; GO:xxxxxxx ; {go\_ref\_db}  
After '%{component\_name} ; GO:xxxxxxx ; {go\_ref\_db}

**parser:** goparser.c -> go\_parser.c

**function:** GODB\_ParseFile( ) -> GO\_LineParser( ) -  
>GO\_AppendRecord( )  
db\_layer: GO\_AppendRecord( ) -  
>GO\_Append\_Reference( )  
GO\_GetRecordByReference( )

**API:** NA

**go\_synonym table**

Last updated: August 5, 2004

**Database:** seqhound  
**Table:** go\_synonym  
**Module:** godb  
**Definition:**  
 MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto incremented id
go_id	int(11)	No	0	GeneOntology ID
go_syn	text	No		GeneOntology Synonym.

MySQL Indexes

Keyname	Type	Field
PRIMARY KEY	INDEX	go_id go_syn(100)
igosynonym_rowid	INDEX	go_id
igosynonym_go_id	INDEX	go_syn

**Observation.:****Organization:** GeneOntology (*ftp://ftp.geneontology.org/pub/go/ontology*)

**Source db:** function.ontology (Molecular function)  
 process.ontology (Biological process)  
 component.ontology (Cellular component)

**Example of source file:** %glycine binding activity ; GO:0016594 ; synonym:Gly binding ;  
 synonym:aminoacetic acid binding ; synonym:aminoethanoic acid  
 binding

\*\*\*go\_id\*\*\*

**description:** Gene Ontology identifier.**example:** 45174**default value:** n/a**source:** The go\_id follows '{function name}; GO:'**parser:** goparser.c -> go\_parser.c

**function:** GODB\_ParseFile() -> GO\_LineParser() -  
 >GO\_AppendRecord()  
 db\_layer: GO\_AppendRecord() -> Go\_Append\_Synonym()

**API:** n/a

\*\*\*go\_syn\*\*\*

**description:** Gene Ontology Synonym Description. This can be a 'synonym' of a 'process' or 'component' name.

**example:** 'Gly binding'

**default value:** n/a

**source:** After '{function\_name} ; GO:xxxxxxx ; synonym:{synonym name}'  
After '{component\_name} ; GO:xxxxxxx ; synonym:{synonym name}'  
After '{process\_name} ; GO:xxxxxxx ; synonym:{synonym name}'  
glutathione dehydrogenase (ascorbate); glutamic acid binding  
%glycine binding activity ; GO:0016594 ;  
synonym:Gly binding ;  
synonym:aminoacetic acid binding ;  
synonym:aminoethanoic acid binding

**parser:** goparser.c -> go\_parser.c

**function:** GODB\_ParseFile( )->GO\_LineParser( )-  
>GO\_AppendRecord( )  
db\_layer: GO\_AppendRecord( )->GO\_Append\_Synonym( )

**API:** n/a

**RPS-BLAST domains (rpsdb) module****domname parser**

---

Note: Not available at this time, as it has not been ported to an ODBC backend. Please go to our ftp site at *ftp://ftp.blueprint.org/pub/SeqHound/RPS/* to download precomputed domname tables.

**Rpsdb parser**

Last updated: August 4, 2004

---

Note: The rpsdb table is precalculated on a cluster and the resulting table is distributed in MySQL format on our ftp site. Therefore, this section is provided for informational purposes only, or for those who would like to build rpsdb tables from their own sequence/domain data; it is not necessary if one wishes simply to include the rpsdb module into their own seqhound instance, in which case they should simply download the precomputed tables.

**domname table**

Last updated: August 4, 2004

**Database:** SeqHound  
**Table:** domname  
**Definition:** Domain information parsed from CDD database.  
**Parser:** The table is populated by the program “DomNameToDB” whose source file is *seqhoun/rps/domname.c*  
The program traverses a directory that contains all the \*.acd files (obtained from NCBI ftp site <ftp://ftp.ncbi.nih.gov/pub/mmdb/cdd/acd.tar.gz>) which contains the CDD ASN.1 records  
**Comments:** Similarly to rpsdb, this table has two API's. The CDD ASN.1 definition seems to be under constant flux and with every new release there are new and revised fields. This in turn requires that we modify the parser functions and update API functions accordingly.

Codebase (for historical purposes)

Column name	Indexed	NULL	Data type	Size	Column Definition
ACCESSION	Yes	No	String	15	CDD id
NAME	Yes	No	String	25	Short domain label
PDB-ID	Yes	Yes	String	10	ID of PDB structure that is used as 3D representative of the domain
ASN1		No	Binary object		The entire CDD ASN.1 record from the source file

MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto incremented id
accession	varchar(15)	No		CDD id
name	varchar(25)	No		Short domain label

pdbid	varchar(12)	No		ID of PDB structure that is used as 3D representative of the domain
asn1	mediumblob	No		The entire CDD ASN.1 record from the source file

## MySQL Indexes

Keyname	Type	Field
idom_rowid	INDEX	rowid
idom_acc	INDEX	accession
idom_name	INDEX	name
idom_pdbid	INDEX	pdbid

## \*\*\*accession\*\*\*

**Description:** This is the domain's Conserved Domain Database (CDD) unique identifier assigned by the CDD group in NCBI. It can also be SMART, Pfam, LOAD or CDD identifier.

**Default Value:** Null

**Source (ASN.1) :** CddIdPtr,

**Parser :** DomNameToDB

**Function:** FillDomNameNode

**More Info:** In early versions of CDD the accession was the Pfam or SMART identifier. Therefore, in some of the comments in the code this field is referred to as the Pfam or SMART identifier. Currently, most of the domains have a unique CDD id but some may not.

**More info:** The API functions return a list. The reason is that a domain label such as SH3 may have a SMART and Pfam entry each stored as a separate CDD entry. Both ID's will be returned.

**API:** SHoundGetDomainIdFromLabel(CharPtr label);



**\*\*\*name\*\*\*****Description:** Domain's short label.**Default value:** Null**source:** This information is obtained from \*.csq file that contains a FASTA description of the domain. The label is parsed from the definition line.**Function:** FillDomNameNode**More info:****API:** string GetDomainLabelFromDomainId(string s); CharPtr LIBCALL  
SHoundGetDomainLabelFromDomainId(CharPtr accession)**\*\*\*pdbid\*\*\*****Description:** A 3D structure representative of the domain.**Default value:** null**source:** pCdd->master3d, part of Cdd record**Function:** FillDomNameNode**More info:****API:** Get3DStructureFromDomainId  
SHoundGetDomain3DStructure**\*\*\*asn1\*\*\*****Description:** This is the entire NCBI CDD ASN.1 structure.**Default value:** Null**source:** Cdd**More info:**

---

---

**Important Note:** The CDD structure contains place holders for describing the domains parent, sibling and child domains. These are domains that are structurally or otherwise related to the domain on the sequence level. The program DomNameToDB collects this information. In the latest release of CDD that was parsed these fields were left empty.

NCBI plans to (or may have) include this information in future releases. This is important information pertaining to the domains that may be included in future versions of the DomName table. Currently, the table has not been expanded with additional fields to hold this information.

**rpsdb table**

Last updated: October 07, 2004

**Database:** SeqHound  
**Table:** rpsdb  
**Definition:** Domain annotation of proteins derived from RPS-BLAST and Conserved Domains Database (CDD).  
**Comments:** The rpsdb is precalculated on a cluster and distributed in MySQL format on our ftp site.  
This table contains two local API versions (C and C++) defined in *rpsdbapi.hpp* and *rpsdbapi.c*. For the most part the same information can be accessed by both API albeit in different forms. However, there may be cases where the two APIs are not identical.

The default e-value cutoff for data in rpsdb is 1

---

Note: The rpsdb table is precalculated on a cluster and the resulting table is distributed in MySQL format on our ftp site. Therefore, this section is provided for informational purposes only, or for those who would like to build rpsdb tables from their own sequence/domain data; it is not necessary if one wishes simply to include the rpsdb module into their own seqhound instance, in which case they should simply download the precomputed tables.

---

Codebase (for historical purposes)

Column_name	Indexed	NULL	Data_type	Size	Column_Definition
GI	Yes	No	Integer	10	Sequence identifier
CDDID	Yes	No	Integer	10	CDD ID (domain ID from CDD)
DOMID	Yes	No	String	12	Domain ID from primary database (Pfam, SMART, COG, KOG or cd)

FROM	No	No	Integer	6	First a.a. position aligned to this domain
ALIGN_LEN	Yes	No	Integer	6	Length of alignment b/w protein and domain
SCORE	No	No	Integer	10	RPS-BLAST score
EVALUE	No	No	Double	15,8	Base 10 log of RPS-BLAST E- value
BITSCORE	No	No	Double	15,8	RPS-BLAST bit score
MISSING_N	No	No	Integer	6	The length of N-terminus residues on the domain that were not aligned
MISSING_C	No	No	Integer	6	The length of C-terminus residues on the domain that were not aligned
NUMDOM	Yes	No	Integer	4	Number of total domains mapped to this protein

## MySQL

Field	Type	Null	Default	Column Definition
rowid	int(11)	No		Auto incremented id
gi	int(11)	No	0	Sequence identifier
cddid	int(11)	No	0	CDD ID (domain ID from CDD)
domid	char(12)	No		Domain ID from primary database (Pfam, SMART, COG, KOG or cd)
rfrom	int(11)	No	0	First a.a. position aligned to this domain
align_len	int(11)	No	0	Length of alignment b/w protein and domain

score	int(11)	No	0	RPS-BLAST score
evaluate	decimal(15,8)	Yes	NULL	Base 10 log of RPS-BLAST E-value
bitscore	decimal(15,8)	Yes	NULL	RPS-BLAST bit score
missing_n	int(11)	No	0	The length of N-terminus residues on the domain that were not aligned
missing_c	int(11)	No	0	The length of C-terminus residues on the domain that were not aligned
numdom	int(11)	No	0	Number of total domains mapped to this protein

## MySQL Indexes

Keyname	Type	Field
irps_rowid	INDEX	rowid
irps_gi	INDEX	gi
irps_cddid	INDEX	cddid
irps_domid	INDEX	domid
irps_len	INDEX	align_len
irps_numdom	INDEX	numdom
irps_gi_e	INDEX	gi evaluate

**Source db:** SeqHound redundant table

**Source program:** RPS-BLAST results

**Parser :** rpsdb.h/c in *seqhound/rps*

\*\*\*gi\*\*\*

**description:** Primary sequence identifier assigned at NCBI

<b>Default value :</b>	0
<b>source:</b>	GI's are collected from Seqhound redund table. Proteins that were not annotated in this table were the hypothetical ORF from RefSeq (XP_XXXXXX) and SWISS-PROT proteins. These proteins are not present in SeqHound (no Bioseq) although their GI's are in redund table. In each redundant group the first GI was used for computing RPS-BLAST (ordinal 1). These proteins are considered to be the best representative of the redundant group. However, some redundant groups may not have the first GI in SeqHound, in those cases the program collects ordinal 2 or higher. The redundant list is collected from redund table using “ <i>redundlist</i> ” program ( <i>seqhound/rps</i> ).
<b>API:</b>	SHoundGetGisByDomainIdAndEval SHoundGetGisByDomainId SHoundGetGisByNumberOfDomains
<b>Comment:</b>	There are two functions in the C version that use Codebase relational query. They should not be used and are there for experimental purposes only.
<b>More info:</b>	<i>seqhound/rps/rpsdb_README.txt</i>
<b>***cddid***</b>	
<b>description:</b>	Conserved Domain Database unique identifier.
<b>Default Value:</b>	none
<b>Source (ASN.1) :</b>	CddIdPtr, can also be collected from CddHitPtr cdd_id field.
<b>Parser :</b>	rpsdb
<b>Function:</b>	RPSDBSHoundRedund2ResultsCallback
<b>More info:</b>	<i>seqhound/rps/rpsdb_README.txt</i>
<b>***domid***</b>	
<b>Description:</b>	Domain identifier from the primary database of origin. These correspond to either Pfam or SMART string identifiers.

**Default Value:** Null  
**source:** Definition field in CddHitPtr structure.  
**Function:** RPSDBSHoundRedund2ResultsCallback

\*\*\*rfrom\*\*\*

**Description:** The index position of the first amino acid in the protein that is aligned with this domain.  
**Default value:** 0  
**source:** Start field in CddHitPtr structure  
**Function:** RPSDBSHoundRedund2ResultsCallback

\*\*\*align\_len\*\*\*

**Description:** The length of sequence alignment between the protein and the domain.  
**Default value :** 0  
**source:** stop- start field values in CddHitPtr.  
**Function:** RPSDBSHoundRedund2ResultsCallback

\*\*\*score\*\*\*

**Description :** RPS-BLAST score parameter  
**Default value:** 0  
**source:** score field in CddHitPtr  
**Function:** RPSDBSHoundRedund2ResultsCallback

\*\*\*evalue\*\*\*

**Description :** Base 10 log of RPS-BLAST evalue score  
**Default value:** 0  
**Source:** evalue field in CddHitPtr  
**Function:** RPSDBSHoundRedund2ResultsCallback

**\*\*\*bitscore\*\*\***

**Description:** RPS-BLAST bit score  
**Default :** 0  
**source:** bitscore field in CddHitPtr  
**Function:** RPSDBSHoundRedund2ResultsCallback

**\*\*\*missing\_n\*\*\***

**Description:** The number of residues on the domain's N-terminus that were not aligned with the protein. The missing length on the N-terminus of the domain.  
**Default:** 0  
**source:** It is collected from DenseDegPtr which is part of SeqAlign structure. This structure is filled up by the RPS-BLAST engine. It contains the collection of aligned segments between the domain and the protein.  
**Function:** RPSDBSHoundRedund2ResultsCallback

**\*\*\*missing\_c\*\*\***

**Description:** The number of missing residues not aligned in the domain's C-terminus.  
**Default:** 0  
**source:** Same as MISSING\_N  
**Function:** RPSDBSHoundRedund2ResultsCallback

**\*\*\*numdom\*\*\***

**Description:** The total number of domains aligned with the protein.  
**Default:** 0  
**source:** The number of entries for the query GI in the rpsdb..  
**Function:** SLRICddCountSeqAligns in rpsdb



**API:**

The above fields are all accessed through a set of calls that retrieve the domain annotation based on different requirements.

SHoundGetDomainsFromGi

SHoundGetDomainsFromGiWithValue

SHoundGetDomainsFromGiListWithValue

## **Molecular Interaction (MI) module**

### **MI-BIND parser**

Last updated October 1, 2004

#### **purpose:**

The Molecular Interaction module is meant to consolidate the interaction data, and associated annotation, from disparate source interaction databases(e.g. BIND, IntAct, MINT, etc). The source data is parsed out of their own unique formats and placed into the MI module tables' data model. This data model has been designed to provide maximum flexibility in terms of the complexity of queries that can be made to it. Additionally, the module adds value to the data by cross-referencing distinct records, regardless of their source databases, to provide information on molecular object redundancy and interaction similarity. The MI module's set of parsers takes records from source interaction modules, parse out their data, and insert that data into the MI tables. Currently, there exists only one such parser, for parsing BIND XML records; parsers for other interaction databases will be developed in the future.

#### **Logic:**

##### **MI-BIND parser:**

This parses the BIND XML records, available on the BIND ftp site. It uses the SAX XML parsing API, in order to achieve maximum parsing speed. In order to foster code reusability, the parser is broken down into three components: The first component parses out the BIND XML records interaction data and places it into a general MI data structure. This data structure is then passed to another component, which is responsible for placing the contents into the MI tables in the database. Another component is then called to do the cross-referencing and redundancy/similarity analysis on the data just processed, and places that additional annotation into the database. This break-down is convenient, because it means that developers of MI parsers for other source interaction databases will only need to replace the first component, the one which parses the data out of the source record, which they can then pass to the already written data feed and cross-referencing components. The parser was written in the Java programming language to take advantage of Java's mature XML processing capabilities, and because a great deal of java source code was available from the open source BIND project for processing BIND XML records.

#### **input files:**

The MI-BIND parser processes BIND records in their native XML format. These are available from the BIND ftp site, in a number of different partitions, each of which can be used interchangeably. In order to import all of BIND into the MI module, one would download the "Divisions" partition from

*ftp://ftp.blueprint.org/pub/BIND/data/divisions/xml/\*.xml*

and then process this list of input files using the MI-BIND parser.

**tables altered:**

MI\_complex2ints

MI\_complex2subunits

MI\_complexes

MI\_dbases

MI\_exp\_methods

MI\_ints

MI\_mol\_type

MI\_obj\_dbases

MI\_obj\_labels

MI\_objects

MI\_record\_types

MI\_refs

MI\_refs\_db

MI\_source

source code location:

The source is contained in the following seqhound java packages:

*org.blueprint.seqhound.parsers.mi*

*org.blueprint.seqhound.parsers.mi.bind*

**config file dependencies:**

*seqhound@blueprint.org*

Version 3.2

### MI.properties file

This file must be present in the same directory as where the MI-BIND parser is being invoked, and must contain the entries:

```
dbDriverName=myDatabaseDriversName
```

```
dbUserName=myDatabaseUserName
```

```
dbPassWord=myDatabasePassword
```

```
dbURL=myDatabaseConnectionURL
```

These contain the settings, which the parser will use to connect to the database management system, which manages the MI data tables. An example entry follows:

```
dbDriverName=com.mysql.jdbc.Driver
```

```
dbUserName=johnsmith
```

```
dbPassWord=johnsmithpassword
```

```
dbURL=jdbc:mysql://dbhostname:dbportnumber/MIdbname
```

**library dependencies:**

Along with the standard java runtime environment, the MI-BIND parser requires one third party java library; this is the open source xerces2 XML parser implementation, available for free from the apache foundation(see <http://xml.apache.org/xerces2-j/download.cgi> for free download instructions).

The xerces jar files *xercesImpl.jar* and *xmlParserAPIs.jar* must be in classpath when executing the parser.

The MI-BIND parser has been tested only with xerces2 java version 2.6.2, but may work with other versions.

**command line parameters:**

The MI-BIND parser takes one command line argument: a file containing a list of filenames of files for the parser to process.

**example use:**

```
java org.blueprint.seqhound.parsers.mi.bind.bmdParse BINDXMLFileList.txt
```

Where *BINDXMLFileList.txt* is a text file containing a newline delimited list of the names of files containing the BIND XML records to be processed; these BIND XML files must be in the same directory. Note that the compiled bind module parser files, as well as the xerces jars, must be in classpath in order for the above command to function.

**associated scripts:**

none

**error and run-time logs:**

runtime and error information is printed to standard output and standard error, respectively. This will likely be changed in a future release, to print to a standard log file.

troubleshooting:

**additional info:**

**MI\_source table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_source  
**Module:** Molecular Interaction(MI)  
**Definition:** Contains information about the source of an interaction record

MySQL

Field	Type	Null	Default	Column Definition
uid	Int(11)	No		Internal MI identifier for record
intcompid	Int(11)	No	0	Internal MI identifier of the interaction, complex or pathway this record refers to
db	Smallint(6)	Yes	NULL	Internal MI identifier of the source db which this record comes from, corresponds to db column of MI_dbases
acc	Varchar(10)	Yes	Null	Source db accession of this record, if it exists
id	Int(11)	Yes	Null	Source db ID of this record, if it exists
type	Smallint(6)	No	Null	Type of this record, either interaction(1), complex(2) or pathway(3)
descr	Text	Yes	Null	Text description of this record from the source db
data_blob	Longblob	Yes	Null	Binary version of original record from source db, if it exists

## MySQL Indexes

data_clob	Longtext	Yes	Null	Text version of original record from source db, if it exists
-----------	----------	-----	------	--

Keyname	Type	Field
Primary	Primary	uid

**Observation:**

**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

\*\*\*uid\*\*\*

**description:**

**example:**

**default value**

**ASN.1 structure:**

**parser:**

**function:**

**API:**

**more info:**

**MI\_ints table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_ints  
**Module:** MI  
**Definition:** Contains information about individual interactions

## MySQL

Field	Type	Null	Default	Column Definition
intid	Int(11)	No		Internal MI id of this interaction
objAid	Int(11)	No	0	Internal MI id of the first object in this interaction
objBid	Int(11)	No	0	Internal MI id of the second object in this interaction
rig	Int(11)	No	0	Internal MI id of the redundant interaction group to which this interaction belongs

## MySQL Indexes

Keyname	Type	Field
Primary	Primary	intid

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND



**MI\_objects table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_objects  
**Module:** MI  
**Definition:** Contains information about the individual objects involved in interactions, complexes and pathways

## MySQL

Field	Type	Null	Default	Column Definition
objid	Int(11)	No		Internal MI identifier for the object which this row describes
type	Smallint(6)	No	0	Internal MI identifier for the molecule type of this object; corresponds to column type of MI_mol_type
db	Smallint(6)	No	0	Internal MI identifier for the source db of this object; corresponds to column db of MI_obj_dbases
id	Int(11)	Yes	Null	Source db id of this object
tax	Int(11)	Yes	Null	NCBI taxonomy id of this object, if applicable
acc	Varchar(20)	Yes	Null	Source db accession of this object
rog	Int(11)	No	0	Internal MI id of the redundant object group to which this object belongs

## MySQL Indexes

Keyname	Type	Field
---------	------	-------

Primary	Primary	objid
---------	---------	-------

**Observation:**

**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_obj\_dbases table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_obj\_dbases  
**Module:** MI  
**Definition:** Table of objects source databases

MySQL

Field	Type	Null	Default	Column Definition
db	Smallint(6)	No		Internal MI identifier for the source db
db_name	Varchar(30)	No		The name of the object source db

MySQL Indexes

Keyname	Type	Field
Primary	Primary	db

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_mol\_types table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_mol\_types  
**Module:** MI  
**Definition:** Molecular types of objects

MySQL

Field	Type	Null	Default	Column Definition
type	Smallint(6)	No		Internal MI identifier for molecules of this type
type_name	Varchar(15)	No		Natural language name for this molecule type

MySQL Indexes

Keyname	Type	Field
Primary	Primary	type

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_dbases table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_dbases  
**Module:** MI  
**Definition:** Table of source databases for interaction records

## MySQL

Field	Type	Null	Default	Column Definition
db	Smallint(6)	No		Internal MI identifier for this source interaction database
db_name	Varchar(30)	No		Natural language name for this source interaction database.

## MySQL Indexes

Keyname	Type	Field
Primary	Primary	db

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_record\_types table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_record\_types  
**Module:** MI  
**Definition:** Table of types for interaction records

## MySQL

Field	Type	Null	Default	Column Definition
type	Smallint(6)	No		Internal MI identifier for this record type
type_name	Varchar(20)	No		Natural language name for this record type(by default, either interaction, complex or pathway).

## MySQL Indexes

Keyname	Type	Field
Primary	Primary	type

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_complexes table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_complexes  
**Module:** MI  
**Definition:** Table of complex and pathway records

## MySQL

Field	Type	Null	Default	Column Definition
compid	Int(11)	No		Internal MI identifier for this complex record
numsubunits	Int(11)	No	0	Number of subunits in this complex

## MySQL Indexes

Keyname	Type	Field
Primary	Primary	compid

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_complex2ints table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_complex2ints  
**Module:** MI  
**Definition:** Mapping of complexes to their component interactions

## MySQL

Field	Type	Null	Default	Column Definition
compid	Int(11)	No	0	Internal MI identifier for this complex
intid	Int(11)	No	0	Internal MI identifier for an interaction which is a component of this complex

## MySQL Indexes

Keyname	Type	Field

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND



**MI\_complex2subunits table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_complex2subunits  
**Module:** MI  
**Definition:** Mapping of complexes to subunits

## MySQL

Field	Type	Null	Default	Column Definition
compid	Int(11)	No	0	Internal MI identifier for this complex
objid	Int(11)	No	0	Internal MI identifier for the subunit object which belongs to this complex

## MySQL Indexes

Keyname	Type	Field

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_complex2subunits table**

Last updated October 4, 2004

**Database:** MI  
**Table:** MI\_complex2subunits  
**Module:** MI  
**Definition:** Mapping of complexes to subunits

## MySQL

Field	Type	Null	Default	Column Definition
compid	Int(11)	No	0	Internal MI identifier for this complex
objid	Int(11)	No	0	Internal MI identifier for the subunit object which belongs to this complex

## MySQL Indexes

Keyname	Type	Field

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_refs table**

Last updated October 6, 2004

**Database:** MI  
**Table:** MI\_refs  
**Module:** MI  
**Definition:** Mapping of MI records to literature references which support it

## MySQL

Field	Type	Null	Default	Column Definition
uid	Int(11)	No	0	Internal MI identifier for the interaction record which this reference is a part of
db	Smallint(6)	No	0	Internal MI identifier for the reference database which this reference is from (eg pubmed or medline)
acc	Varchar(15)	Yes	Null	Accession of this reference from source reference database
id	Int(11)	Yes	Null	ID of this reference from the source reference database
method	Smallint(6)	No	0	Internal MI identifier for the experimental method used in the referenced experiment

## MySQL Indexes

Keyname	Type	Field

**Observation:**

<b>Source org:</b>	Blueprint
<b>Source file:</b>	BIND division files
<b>FTP script:</b>	N/A
<b>Parser:</b>	MI-BIND

**MI\_refs\_db table**

Last updated October 6, 2004

**Database:** MI  
**Table:** MI\_refs\_db  
**Module:** MI  
**Definition:** Mapping of internal MI reference database Ids to source reference databases

## MySQL

Field	Type	Null	Default	Column Definition
db	Smallint(6)	No		Internal MI identifier for this reference database
db_name	Varchar(15)	Yes	Null	The natural language name of the reference source database

## MySQL Indexes

Keyname	Type	Field
Primary	Primary	db

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_exp\_methods table**

Last updated October 6, 2004

**Database:** MI  
**Table:** MI\_exp\_methods  
**Module:** MI  
**Definition:** Mapping of internal experimental method identifiers to their descriptions

## MySQL

Field	Type	Null	Default	Column Definition
method	Smallint(6)	No		Internal MI identifier for this experimental method
method_descr	Varchar(40)	Yes	Null	Natural language name of the experimental method

## MySQL Indexes

Keyname	Type	Field
Primary	Primary	method

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND

**MI\_obj\_labels table**

Last updated October 6, 2004

**Database:** MI  
**Table:** MI\_obj\_labels  
**Module:** MI  
**Definition:** Mapping of molecular objects to their free form labels

## MySQL

Field	Type	Null	Default	Column Definition
uid	Int(11)	No	0	Internal MI identifier for the interaction/complex/pathway record which assigns this label to this object
objid	Int(11)	No	0	Internal MI identifier for the molecular object being labeled
label	Text	Yes	Null	Free form label given to this molecular object by this interaction/complex/pathway record

## MySQL Indexes

Keyname	Type	Field

**Observation:**  
**Source org:** Blueprint  
**Source file:** BIND division files  
**FTP script:** N/A  
**Parser:** MI-BIND





## **Text mining module**

### **Overview:**

The SeqHound text mining module helps researchers locate mentions and co-mentions of biologically related entities in the scientific literature.

At the time of writing this is limited to finding protein mentions in PubMed abstracts. The module however has been designed to be extensible to small molecules, complexes and even biological concepts in both abstract and full-text articles. Each of the steps in the process may be scored by multiple methods that can be developed internally or by external developers.

### **mother parser**

Note that a number of the tables in the text module are created at the same time as the creation of the other core tables (see core.sql) and some of these tables are populated by the mother parser (see table descriptions below). Mother, is used to retrieve protein names and synonyms from the RefSeq database, so these tables are created at the same time as other core module tables. The mother parser is described under the core module section above. Parsers specific to the text module are described below.

**text searcher parser**

last updated February 25<sup>th</sup>, 2005

**purpose:**

The text update parser and related programs are used to collect bionames, search against a literature database, and then investigate co-mentions of bioentities in the literatures. The co-mentions of bioentities are scored using pattern recognition and statistical machine learning methods to look for potential biophysical interactions between bioentities. The text mining module tables are updated daily.

**logic:**

Most of the update logic is implemented in the Text.pm Perl module and other scoring programs. The text update parser calls the functions in Text.pm and other Perl modules. The text mining module depends on the CORE module to generate and update names of proteins and on the MyMED in house literature database. These resources are updated daily. The steps taken to update the text-mining module are as follows:

Step 1: Collect bioentity names from the lexicon

Step 2: Formulate searches; collect search results and scores

Step 3: Collect co-occurrences of names and scores

Step 4: Summarize evidence for each pair of bioentities and scores

**module: text****input files:**

Latest Medline release: [ftp://ftp.ncbi.nih.gov/nlmdata/.medlease/\\*.xml.gz](ftp://ftp.ncbi.nih.gov/nlmdata/.medlease/*.xml.gz)

Latest PubMed Central release: <ftp://ftp.ncbi.nlm.nih.gov/pub/pmc/>

Latest Biomed Central release: <ftp://ftp.biomedcentral.com/>

Latest NRC LitMiner SVM release: <http://ii200.iit.nrc.ca/~martinj/>

British National Corpus: <ftp://ftp.itri.bton.ac.uk/bnc/>

Moby project English word list: <http://www.dcs.shef.ac.uk/research/ilash/Moby/>

PubMed help stop word list: <http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html#Stopwords>

Smart English stop word list: <ftp://ftp.cs.cornell.edu/pub/smart/english.stop>

### **tables altered:**

text\_bncorpus, text\_db, text\_doc, text\_docscore, text\_doctax, text\_englishdict, text\_evidence, text\_evidencescore, text\_method, text\_namepair, text\_namepairresult, text\_organism, text\_pattern, text\_point, text\_pointscore, text\_result, text\_resultscore, text\_rng, text\_rngscore, text\_search, text\_searchscore, text\_stopword

### **source code location:**

*slri/seqhound/text/text\_update.pl*

### **config file dependencies:**

*slri/seqhound/text/text.ini*

### **command line parameters:**

Typing “**./text\_update.pl -**” or “**perl text\_update.pl -**” at the command line while in the directory where text\_update.pl resides will return a list of command line parameters and default settings.

text\_update.pl arguments:

-r redo all name searches against new insertions from MyMED using time stamp [T/F] optional

-t taxonomy ids for organism to be searched, seperated by comma. If specified in command line, only provided organism(s) will be searched. Otherwise, a default list of taxids stored in text.ini file will be searched.  
Optional

-u update target table only, default is all. Optional

**example use:**

For example:

```
>./text_update.pl -r F -t 4932,10090,9606 -u text_searchscore
```

**associated scripts:**

*slri/seqhound/text/text.sql* Data Definition file for text mining module tables.

*slri/seqhound/text/text\_dump.sh* Mysql dump script for portable tables.

*slri/seqhound/text/text\_create.sh* Script used to create Text database, the starting point.

*slri/seqhound/text/text\_regex.pl* Script used to score evidence using regex patterns.

*slri/seqhound/text/Text.pm* contains most functions that will be using for update.

*slri/seqhound/text/Pattern.pm* Perl module used to represent regular expression objects.

*slri/seqhound/text/Tee.pm* Perl module used to branch the output to different outputs.

*slri/seqhound/text/text\_updatecron.sh* Text module daily update cron script.

*slri/seqhound/text/myeutils.pl* Entrez util script used for comparing results between MyMED and entrez searches.

**error and run-time logs:**

Errors and runtime logs will be directed to file specified in text.ini , current default log file name is text\_update.log

Daily update log will also be send to email account specified in text.ini file

**troubleshooting:**

Check the email message sent by text\_update.pl to see if there is any error during update. Consult update log file to look for detail problem.

**additional info:**

All text mining module tables are in small cases, and are prefixed with "text\_". Tables with auto incremented rowid have a primary key "id" as default. When referencing primary key in other table, the field name in the referencing table will be the referenced table name plus id.

**yeastnameparser.pl parser**

Last updated September 27, 2004

**purpose:**

The yeastnameparser extracts names from the SGD file *SGD\_features.tab*. Only names that belong to yeast records that are already in RefSeq are added, as determined using the DBXRef module. This means that this parser **MUST** be run after the DBXRef parsers.

It is not necessary to run this parser, if yeast names are not desired.

**logic:**

The yeastnameparser reads through each record in *SGD\_features.tab*. It searches for a refseq cross reference for each record. If one is found, then the parser gets the relevant bioentityid and checks all names for that bioentityid.

If the name does not already exist in the database, then it is added and the db field is set to “sgd” and the access field is set to the SGDID. The action field is set to 1, ADD and the current data is written to actiondate.

Existing names in the database are also compared to the file. Names which are no longer present in the yeast file are marked as deleted (action =2) and the current data is written to actiondate. If the name already exists in refseq, then a SECONDREFS record is filled out for the yeast record.

module: names

input files:

*SGD\_features.tab* from

*ftp://genome-ftp.stanford.edu/pub/yeast/data\_download/chromosomal\_feature/*

**tables altered:**

bioentity, bioname, secondrefs

**source code location:**

*slri/seqhound/names/yeastnameparser.pl*

**config file dependencies:**

The relevant configuration files are:

*.intrezrc* and *.odbc.ini* should be set up as described above for seqhound. The values will be read by *shconfig.pm*, which should be located in the same directory as *yeastnameparser*.

**command line parameters:**

None.

**example use:**

**perl yeastnameparser.pl**

**associated scripts:**

The program *yeastnamecron.pl* can be used for both the initial read of *SGD\_features.tab* and updates of the file. *yeastnamecron.pl* checks whether *SGD\_features.tab* needs updating, downloads it and calls *yeastnameparser.pl*. *SGD\_features.tab* is updated weekly by SGD.

**error and run-time logs:**

*yeastnameparser* writes errors to a file called *yeastname.log*. Updates are written to a file called *yeastupdate.log* as a tab delimited file where the fields are: name, sgdid, bionameid and field. Additions are written to a file called *yeastadd.log* as a tab delimited file where the fields are: name, sgdid bioentityid and field.

**troubleshooting:**

**additional info:**

**text\_bioentity table**

Last updated Febuary 15, 2004

**Database:** SeqHound

**Table:** text\_bioentity

**Module:** text (note mother parser is part of the core module)

**Definition:** A bioentity refers to any biological object with names that may be used in the literature to refer to these objects. Currently, all bioentities are proteins from RefSeq. This table tells us which database contains the primary record for this bioentity, the accession and the field which refers to this bioentity. At the moment, all bioentities are obtained from RefSeq. RefSeq was chosen because it represents a high quality database of non-redundant records. The intention of this part of the core is to collect a non-redundant list of biological objects and the names that are used in written language to refer to them by.

**Source file:** \*.bna.gz from *ftp://ftp.ncbi.nih.gov/refseq/release/complete*

**FTP script:** *asnftp.pl*

**Parser:** mother



## text\_bioentity table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(11)	no		Auto incremented id. A unique identifier for this bioentity.	1	mother parser autoincrements this field	SHoundBioentityIdFromGi SHoundBioentityIdFromAcc SHoundBioentityIdListFromBionameAndTaxId
bioentitytypeid	int(11)	No	0	Bioentity type (for example, molecule type). Id from the bioentitytype table.	1	mother parser (see GetType()) parsing and see bioseq->mol	SHoundBioentityFromBioentityId
db	varchar(15)	No		The primary database where this bioentity was found.	ref	mother parser inserts "ref"	SHoundBioentityFromBioentityId
access	varchar(20)	No		Accession in the primary database. Any alphanumeric identifier used by the primary database.	NP_858066	mother parser (see FillBionameDB()) and see bioseq-->seqid	SHoundBioentityFromBioentityId
identifier	int(11)	Yes		Numeric identifier for this bioentity in the primary database; for example an NCBI Gene Info identifier (GI). This field is not required.	31982991	mother parser (see GetGI) see bioseq-->seqid (choice 12)).	SHoundBioentityFromBioentityId
fieldtypeid	int(11)	No		A number that represents the field from which this bioentity was derived. This is the id from the fieldtype table. For example, 1 indicates the ASN.1 path "seqentry/seqset/bioseq"	1	mother parser (see ASN.1 path)	SHoundBioentityFromBioentityId

## text\_bioentity indices

Keyname	Type	Field
PRIMARY	PRIMARY	access
ibioe_id	INDEX	id
ibioe_type	INDEX	bioentitytypeid
ibioe_identifier	INDEX	identifier

**text\_bioname table**

Last updated February 15, 2005

<b>Database:</b>	SeqHound
<b>Table:</b>	text_bioname
<b>Module:</b>	text (note that the mother parser is part of the core module)
<b>Definition:</b>	This table holds names for bioentities. A Bioname is some “name” applied to some bioentity.
<b>Source org:</b>	NCBI
<b>Source file:</b>	<i>*.bna.gz from <a href="ftp://ftp.ncbi.nih.gov/ncbi/refseq/release/complete">ftp://ftp.ncbi.nih.gov/ncbi/refseq/release/complete</a></i>
<b>FTP script:</b>	<i>asnftp.pl</i>
<b>Parser:</b>	mother

## text\_bioname table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(11)	No		A unique identifier for this bioentity-name pair.	1	mother parser auto increments this column	SHoundBionameListFromBioentityId
timestamp	datetime	Yes	NULL			mother parser yeastnameparser	
bioentityid	int(11)	No	0	The id of the bioentity to which this name refers.	1	mother parser yeastnameparser	SHoundBionameListFromBioentityId
name	text	No		The name.	2- isopropylmal ate synthase	mother parser yeastnameparser	SHoundBionameListFromBioentityId
nametypeid	int(11)	No	0	Type of name. For example, protein name (1) or gene name (2). This is the id from the nametype table.	1	mother parser yeastnameparser	SHoundBionameListFromBioentityId
db	varchar(15)	No		The database in which this name was found.	ref	mother parser yeastnameparser	SHoundBionameListFromBioentityId
access	varchar(20)	No		The accession of the record in which this name was found.	NP_047187	mother parser yeastnameparser	SHoundBionameListFromBioentityId
identifier	int(11)	Yes	NULL	The identifier of the record in which this name was found; for example a Gene Info identifier (GI).	10954458	mother parser yeastnameparser	SHoundBionameListFromBioentityId
fieldtypeid	int(11)	No	0	The field of the record in which this name was found. This is the id from the fieldtype table; for example, 5 indicates the ASN.1 path "seqentry/seqset/bioseq/description/"	5	mother parser yeastnameparser	SHoundBionameListFromBioentityId

official	int(11)	Yes	0	Is this an official name? 1=Yes, 2=N0	1	yeastnameparser	SHoundBionameListFromBioentityId
deprecated	int(11)	Yes	0	Has this name been deprecated? Not used at present.	0		SHoundBionameListFromBioentityId
datedeprecated	datetime	Yes	000000000 00000	The date the name was deprecated.	00000000000 000	mother parser yeastnameparser	SHoundBionameListFromBioentityId
ruleid	int(11)	Yes	NULL	What rule was used to construct this name? This is the id from the rules table. For example, 1 indicates that A gene name is being used to refer to a protein.	1	mother parser yeastnameparser	SHoundBionameListFromBioentityId
action	char(1)	Yes	A	What was the last action taken on this record. A=ADD, D=DELETE.		mother parser yeastnameparser	
actiondate	datetime	Yes	000000000 00000	Date of the last action.	2004-08-20 01:30:08	mother parser yeastnameparser	

## text\_bioname indices

Keyname	Type	Field
ibioname_id	INDEX	id
ibioname_identifier	INDEX	identifier
ibioname_access	INDEX	access
ibioname_bioentityid	INDEX	bioentityid
ibioname_nametypeid	INDEX	nametypeid
ibioname_official	INDEX	official
ibioname_deprecated	INDEX	deprecated
ibioname_ruleid	INDEX	ruleid
ibioname_action	INDEX	action
ibioname_actiondate	INDEX	actiondate

**text\_secondrefs table**

Last updated February 15, 2005

<b>Database:</b>	SeqHound
<b>Table:</b>	text_secondrefs
<b>Module:</b>	core
<b>Definition:</b>	Additional references for sources of bionames.
<b>Source org:</b>	SGD
<b>Source file:</b>	<i>SGD_features.tab</i> from <a href="ftp://genome-ftp.stanford.edu/pub/yeast/data_download/chromosomal_feature/">ftp://genome-ftp.stanford.edu/pub/yeast/data_download/chromosomal_feature/</a>
<b>FTP script:</b>	<i>yeastnamecron.pl</i>
<b>Parser:</b>	<i>yeastnameparser.pl</i>

## text\_secondefs table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(11)	No		id of the second reference. Mysql auto-increment column.	1	yeastnameparser will Auto-increment this column	
timestamp	datetime	Yes	NULL			yeastnameparser	
bionameid	int(11)	No	0	id of the name that this reference refers to.	1	yeastnameparser retrieves this from text_bioname table	
db	varchar(15)	No		Database in which this reference is found.	sgd	yeastnameparser (see AddSeconfRef())	
access	varchar(20)	Yes	NULL	Accession of the record that refers to the name.	S000033	yeastnameparser (see AddSeconfRef())	
fieldtypeid	int(11)	No	0	Identifies the field in the record where this name was found.	101	yeastnameparser (see AddSeconfRef())	



text\_secondrefs indices

Keyname	Type	Field
isecondrefs_id	INDEX	id
isecondrefs_bionameid	INDEX	bionameid
isecondrefs_dbsearch	INDEX	access, db
isecondrefs_field	INDEX	field

**text\_bioentitytype table**

Last updated February 15, 2005

**Database:** SeqHound  
**Table:** text\_bioentitytype  
**Module:** text (note that this table is currently defined when the core module is created)  
**Definition:** Look up table that stores bioentity types.  
**Source org:** Blueprint  
**Source file:** *core.sql*  
**FTP script:** NA  
**Parser:** NA

## text\_bioentitytype table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(11)	No		Identifier for a bioentitytype. Mysql auto-increment column.	1	core.sql	
type	varchar(80)	No		The molecule type.	protein	core.sql	

## text\_bioentitytype indices

NA

**text\_fieldtype table**

Last updated February 15, 2005

**Database:** SeqHound  
**Table:** text\_fieldtype  
**Module:** text (note that this table is defined when the core module is created)  
**Definition:** Look up table that stores the field that contains the name.  
**Source org:** Blueprint  
**Source file:** This is a look up table that is filled by *core.sql*  
**FTP script:** NA  
**Parser:** NA

## text\_fieldtype table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		Identifier for a fieldtype.	1	core.sql	
pathtofield	varchar(80)	No		The field that contains the name. For an ASN.1 record, this is the ASN.1 "path" to the field that contains the name. For a flat file, this is of the form "Column #:Column Name".	seqentry/seqset/bioseq/seq-annot/seqfeat-gene/syn	core.sql	

## text\_fieldtype indices

NA

**text\_nametype table**

Last updated February 15, 2005

**Database:** SeqHound  
**Table:** text\_nametype  
**Module:** text (note that this table is defined when the core module is created)  
**Definition:** Look up table that stores name types.  
**Source org:** Blueprint  
**Source file:** This is a look up table that is filled by *core.sql*  
**FTP script:** NA  
**Parser:** NA

## text\_nametype table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(11)	No		Identifier for a nametype. Mysql auto-increment column.	1	core.sql	
type	varchar(880)	No		The type of name.	protein	core.sql	

## text\_nametype indices

NA

**text\_rules table**

Last updated September 30, 2004

**Database:** SeqHound  
**Table:** text\_rules  
**Module:** text (note that this table is defined when the core module is created)  
**Definition:** Look up table that stores rules for generating names.  
**Source org:** Blueprint  
**Source file:** This is a look up table that is filled by *core.sql*  
**FTP script:** NA  
**Parser:** NA

## text\_rules table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		Identifier for a rule. Mysql auto-increment column.	1	core.sql	
type	varchar(80)	No		The rule.	Use gene name for protein.	core.sql	

## text\_rules indices

NA

**text\_db table**

Last updated February 15, 2005

**Database:** seqhound

**Table:** text\_db

**Module:** text

**Definition:** Lookup table that lists biomedical literature databases used by the text module.

**Source file:**

**Source org:** Blueprint

**FTP script:** NA

**Parser:** NA

**text\_db table**

Field	Type	Null	Default	Column Definition	Example	Source	API
dbid	int(11)	No		Auto incremented id	1	text.sql	
name	varchar(80)	No		name of database	PubMed	text.sql	

**text\_db indices**

Keyname	Type	Field
PRIMARY	PRIMARY	dbid

**text\_doc table**

Last updated February 16, 2005

<b>Database:</b>	seqhound
<b>Table:</b>	text_doc
<b>Module:</b>	text
<b>Definition:</b>	This table lists the accession ids from each literature database and assigns them an internal document id.
<b>Source file:</b>	<u><a href="ftp://ftp.nlm.nih.gov/nlmdata/.medlease/medline*.xml.gz">ftp://ftp.nlm.nih.gov/nlmdata/.medlease/medline*.xml.gz</a></u> <u><a href="ftp://ftp.ncbi.nlm.nih.gov/pub/pmc/">ftp://ftp.ncbi.nlm.nih.gov/pub/pmc/</a></u>
<b>Source org:</b>	NLM MEDLINE and other biomedical literature databases
<b>FTP script:</b>	<i>slri/medline/updates/updatecron.sh</i> <i>slri/medline/pubmedcentral/pmc_updatecron.sh</i>
<b>Parser:</b>	text_update.pl

## text\_doc table

Field	Type	Null	Default	Column Definition	Example	Source	API
docid	int(11)	No		auto incremented identifier	1077882		
dbid	int(11)	No	0	literature database id	1		
accession	int(11)	No	0	accession in literature database such as a PubMed identifier (PMID)	1077882	Medline Xpath: “/MedlineCitation/PMID” Or PMC Xpath: /art/ui[@type="pmid"]	
status	char(10)	Yes	NULL	insert, delete or update	NULL		

## text\_doc indices

Keyname	Type	Field
id	PRIMARY	id
dbid	INDEX	dbid
accession	INDEX	accession



**text\_docscore table**

Last updated September 28, 2004

<b>Database:</b>	seqhound
<b>Table:</b>	text_docscore
<b>Module:</b>	text
<b>Definition:</b>	This table lists scores for documents. Multiple scores (from different scoring methods) may be listed for each document.
<b>Source file:</b>	
<b>Source org:</b>	NRC
<b>FTP script:</b>	<i>wget -P ~/nrc -m <a href="http://ii200.iit.nrc.ca/~martinj/slri/text/nrc.sh">http://ii200.iit.nrc.ca/~martinj/slri/text/nrc.sh</a></i>
<b>Program:</b>	LitMiner : <i><a href="http://textomy.iit.nrc.ca/cgi-bin/bindpresent.cgi?qry=10747882">http://textomy.iit.nrc.ca/cgi-bin/bindpresent.cgi?qry=10747882</a></i>
<b>Notes:</b>	The current method for scoring a document is an SVM (support vector machine) classifier implemented by National Research Council's Joel Martin and Berry deBruijn. Detailed information can be found in the prebind publication: <i><a href="http://www.pubmedcentral.nih.gov/articlerender.fcgi?tool=pubmed&amp;pubmedid=12689350">http://www.pubmedcentral.nih.gov/articlerender.fcgi?tool=pubmed&amp;pubmedid=12689350</a></i>

## text\_docscore table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		foreign key from doc table. See docid	25		
methodid	int(11)	No		scoring method identifier. See the text_method table. For example, 1 indicates an SVM trained to recognize papers describing interaction data.	1		
score	double	No	0	score	-0.852876574	<a href="http://ii200.iit.nrc.ca/~martinj/">http://ii200.iit.nrc.ca/~martinj/</a>	

## text\_docscore indices

Keyname	Type	Field
PRIMARY	PRIMARY	(docid, methodid, score)
methodid	INDEX	methodid
score	INDEX	score

**text\_evidence** table

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_evidence

**Module:** text

**Definition:** This table assigns a unique identifier to each co-occurrence of two bionames in the same document. A bioname points to a specific name bioentity pair in the bioname table (this pair is repeated here). This is to be distinguished from a pair of names that co-occur in a document; this pair of names is identified by a namepairid and does not make any reference to a specific pair of bioentities. This evidence is in support of some point and is based on some name pair.

**Program:**

text\_evidence table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(11)	No		A unique identifier for this piece of evidence.	1	Auto incremented identifier	
docid	int(11)	No	0	the document identifier where this evidence occurs	325387		
resultidA	int(11)	No	0	search result identifier for bioname A	2200		
bioentityida	int(11)	No	0	identifier for bioentity A	107564		
nama	char(80)	No		name referring to A	LEU1		
resultidB	int(11)	No	0	search result identifier for bioname B	9513		
bioentityidb	int(11)	No	0	identifier for bioentity B	109911		
namb	char(80)	No		name referring to B	HIS4		
pointid	int(11)	Yes	NULL	POTential INTERaction between two bioentites that this evidence supports	2		
namepairid	int(11)	Yes	NULL	corresponding namepairid on which this evidence is based	10634		
state	smallint(6)	Yes	NULL	book keeping	0		

text\_evidence indices

Keyname	Type	Field
PRIMARY	PRIMARY	evidenceid
docid	INDEX	docid
resultida	INDEX	resultidA
resultidb	INDEX	resultidB

**text\_evidencescore table**

Last updated February 16, 2005

**Database:** seqhound

**Table:** text\_evidencescore

**Module:** text

**Definition:** This table assigns a score to a piece of evidence. See evidence table. Multiples cores may be assigned to one evidence if different methods were used. Currently, this table stores potential protein-protein interaction scores for each co-occurrence of bionames. The current scoring method uses a set of manually collected regular expression patterns to identity interactions. Detailed information can be found at: *<ftp://ftp.blueprint.org/pub/BIND/PreBIND/README>*

**Program:**

text\_evidencescore table

Field	Type	Null	Default	Column Definition	Example	Source	API
evidenceid	int(11)	No	0	see text_evidence table	8		
methodid	int(11)	No	0	scoring method identifier (see text_method table); for example, 2 indicates a method that uses regular expressions to detect bioentities that are physically interacting	2		
score	double	No	0	the score	0.63		

text\_evidencescore indices

Keyname	Type	Field
PRIMARY	PRIMARY	(evidenceid, methodid, score)
methodid	INDEX	methodid
score	INDEX	score

**text\_method table**

Last updated March 15, 2005

<b>Database:</b>	seqhound
<b>Table:</b>	text_method
<b>Module:</b>	text
<b>Definition:</b>	This table assigns a unique identifier to each of the multiple scoring schemes used in the SeqHound text mining module.
<b>Source:</b>	This file is hand-edited.
<b>Source Org.</b>	Blueprint



text\_method table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No	0	Auto incremented identifier	1		
type	varchar(30)	No		This field describes the type of score that is generated by the method. It will be one of searchscore, docscore, resultscore, evidencescore or pointscore.	searchscore		
hypothid	int(11)	Yes	0	corresponding hypothesis identifier. See the text_hypoth table.	23		
hypoth	text	Yes	NULL	a hypothesis that this method attempts to support or refute.	This document describes biophysical interaction data for some molecule(s).		
method	text	Yes	NULL	a more detailed description of the method with pointers to more details about the method and its performance.	A support vector machine was trained to recognize abstracts containing biophysical interaction data. See PubMed Identifier 12689350 for more details.		
positive	text	Yes	NULL	the value (range) of scores corresponding to the hypothesis being found TRUE	>0		
negative	text	Yes	NULL	the value (range) of scores corresponding to the hypothesis being found FALSE	<0		

undecided	text	Yes	NULL	the value (range) of scores corresponding to the hypothesis being found UNDECIDED	0		
implemented	enumerated	Yes	NULL	has this method been implemented	NO		
assume	text	Yes	NULL	the score value that can be assumed if the method has been implemented but no score is found	>0		
script	text	Yes	NULL	the script or program that implements the method	text_search.pl		

text\_method indices

Keyname	Type	Field
methodid	PRIMARY	methodid

**text\_point table**

Last updated February 16, 2005

**Database:** seqhound

**Table:** text\_point

**Module:** text

**Definition:** A POINT represents two bioentities for which some POTential INTERaction may occur in the literature. Each POINT may be supported by multiple pieces of evidence in the evidence table.

**Program:**

## text\_point table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		auto incremented identifier	1		
bioentityidA	int(11)	No	0	bioentity identifier	110070		
bioentityidB	int(11)	No	0	bioentity identifier	1268438		
state	small int	Yes	NULL	book keeping	0		

## text\_point indices

Keyname	Type	Field
tid	PRIMARY	id
bioentityida	INDEX	bioentityida
bioentityidb	INDEX	bioentityidb

**text\_pointscore table**

Last updated February 16, 2005

**Database:** seqhound

**Table:** text\_pointscore

**Module:** text

**Definition:** This table lists scores for potential interactions. These scores may be viewed as a summary of the scores for all the pieces of evidence that support this POINT. A POINT may have multiple scores that are generated by multiple methods.

**Program:**

## text\_pointscore table

Field	Type	Null	Default	Column Definition	Example	Source	API
pointid	int(11)	No	0	a unique identifier for some POINT (see text_point table)	11		
methodid	int(11)	No	0	an identifier for the method used to generate this score for the POINT (see text_method table).	3		
score	double	No	0	the score	0.63		

## text\_pointscore indices

Keyname	Type	Field
PRIMARY	PRIMARY	(pointid, methodid, score)
methodid	INDEX	methodid
score	INDEX	score
pointid	INDEX	pointid

**text\_result table**

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_result

**Module:** text

**Definition:** This table stores all the search results (document ids) for performed searches (see text\_search table). Position default is 0, which means that the bioname appears somewhere in the document without specifying exactly where or how many times.

**Program:**

text\_result table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(11)	No		auto incremented identifier	1		
searchid	int(11)	No	0	identifies the search that generated this result	1		
docid	int(11)	No	0	the document identifier where the name was found	7874750		
positionid	int(11)	No	0	the position id in the document where the name appears (0 indicates no specified position)	0		
state	smallint(6)	Yes	NULL	book keeping	0		

text\_result indices

Keyname	Type	Field
tid	PRIMARY	id
searchid	INDEX	searchid
docid	INDEX	docid
positioned	INDEX	positionid

**text\_resultscore table**

Last updated February 16, 2005

**Database:** seqhound

**Table:** text\_resultscore

**Module:** text

**Definition:** This table holds scores for search results; i.e., is the searched-for bioentity really mentioned in the document. This table might be used to store disambiguation scores for search results. Because a bioname can refer to many different bioentities, an algorithm (some method) may be used to determine which bioentity a name occurrence refers to. This table might also hold the results of methods that disambiguate bioentities that have English words as names.

**Program:**



## text\_resultscore table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No	0	Auto incremented identifier	1		
methodid	int(11)	No	0	identifier for the method used to score this result (see text_method table)	3		
score	double	No	0	the score	-1		

## text\_resultscore indices

Keyname	Type	Field
PRIMARY	PRIMARY	(resultid, methodid, score)
resultid	INDEX	resultid
methodid	INDEX	methodid
score	INDEX	score

**text\_search table**

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_search

**Module:** text

**Definition:** This search table is generated using the text\_bioname table. A search is minimally composed of a name that is used to look for some bioentity (listed in this table) using some method. Currently all bioname items with nametypeid=2 and a taxid that can be found in the taxgi table and are inserted into this search table for MyMedline searching.

**Program:**

## text\_search table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(11)	No		auto incremented identifier	1		
bioentityid	int(11)	No	0	bioentity being searched for	106945		
bionameid	int(11)	No	0	bionameid used to search for bioentity	414737		
name	char(80)	No		name used to find mention of bioentity	AI1		
taxid	int(11)	No	0	taxonomy identifier for bioentity (if applicable)	4932		
rngid	int(11)	Yes	NULL	redundant name group identifier	70513		
methodid	int(11)	No		method used to search for the bioentity mention using name	1		
searched	datetime	No	0000-00-00 00:00:00	date and time that search was last performed	2004-12-01 10:12:22		
results	int(11)	Yes	NULL	number of search results returned	17		

## text\_search indices

Keyname	Type	Field
PRIMARY	PRIMARY	searchid
bioentityid	INDEX	bioentityid
bionameid	INDEX	bionameid

taxid	INDEX	taxid
searched	INDEX	searched

**text\_searchscore table**

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_searchscore

**Module:** text

**Definition:** This table holds some score that may be used to determine if a given search strategy WILL BE informative if performed or its results are likely TO BE informative if examined. This score might also be used to determine whether a search strategy is to be performed at all or if some search strategy is best left until more informative searches have been informed. For example, if the name to be used is an English word, the search may be scored so as to skip this search. Multiple methods (and their scores) may be applied to a single search.

**Program:**

## text\_searchscore table

Field	Type	Null	Default	Column_Definition	Example	Source	API
searchid	int(11)	No	0	identifies a search strategy (see text_search table)	1		
methodid	int(11)	No	0	identifies the method used to score the search strategy			
score	double	No	0	the score			

## text\_searchscore indices

Keyname	Type	Field
PRIMARY	PRIMARY	(searchid, methodid, score)
searchid	INDEX	resultid
methodid	INDEX	methodid
score	INDEX	score

**text\_rng table**

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_rng (redundant name group)

**Module:** text

**Definition:** This table groups together bionames that have equivalent names (i.e. homonyms). This table facilitates searching by reducing the eliminating redundant searches for the same string in the document collection. It is thus an intermediate table in the process of creating the text\_result table.

**Program:**

## text\_rng table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		auto incremented identifier. This is a unique identifier for the redundant name group identifier.	8		
name	char(80)	No		the homonym represented by this group	SRB8		
searched	datetime	No	0000-00-00 00:00:00	when this name was last used to search	2004-12-01 12:02:23		
results		Yes	NULL	the number of documents returned by this search	0		
status	char(20)	Yes	NULL	book keeping	searched		

## text\_rng indices

Keyname	Type	Field
Id	PRIMARY	id
Name	Unique	Name
Searched	INDEX	Searched
Results	INDEX	Results
Status	INDEX	status



**text\_rngresult table**

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_rngresult

**Module:** text

**Definition:** This table is an intermediate step in one method for searching for mentions of protein names in text. The table stores search results for redundant name groups. These results are combined with the text\_doctax table to generate the final text\_results table.

**Program:**

**Note:** This is an intermediate table and is not distributed as part of SeqHound.

## text\_rngresults table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		Auto incremented identifier.	1		
rngid	int(11)	No	0	a redundant name group identifier. See the text_rng table.	2		
docid	int(11)	Yes	NULL	a document identifier where this name appears	45549		
pmid	int(11)	Yes	NULL	the corresponding PubMed identifier where this name appears	45549		
state	int(11)	No	0	book keeping	0		

## text\_rngresults indices

Keyname	Type	Field
Id	PRIMARY	Id
Rngid	INDEX	Rngid
Docid	INDEX	Docid
Pmid	INDEX	Pmid
State	INDEX	State

**text\_doctax table**

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_doctax

**Module:** text

**Definition:** This table keeps a list of the organisms (by taxon identifiers) that are described in a document.

**Program:**

## text\_doctax table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		Auto incremented identifier. A unique identifier.	1		
docid	int(11)	No	0	document identifier (see text_doc table).			
taxid	int(11)	No	0	organism described in this document (listed by NCBI taxonomy database identifier)			

## text\_doctax indices

Keyname	Type	Field
Id	PRIMARY	Id
Docid	INDEX	Docid
Taxid	Index	taxid

**text\_organism table**

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_organism

**Module:** text

**Definition:** This table keeps a list of the MESH terms that are used to identify the presence of an organism in a PubMed abstract.

**Program:**

**Source Org:** Blueprint

**Source file:**

**Note:**

## text\_organism table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		auto incremented identifier	1		
taxid	int(11)	No	0	NCBI taxonomy identifier	4932		
mesh	timestamp	No		mesh term found in abstracts that describe this organism	Saccharomyces cerevisiae		
searched	timestamp	Yes	CURRENT_TIMESTAMP	time that this search was last completed	2005-02-15 15:58:38		
results	int(11)	Yes	NULL	number of documents returned	NULL		
bioentities	int(11)	Yes	NULL	number of bioentities for this organism	NULL		
bionames	int(11)	Yes	NULL	number of names for this organism	NULL		
maxbionameid	int(11)	Yes	NULL	maximum bioname identifier	NULL		
lastupdate	timestamp	Yes	0000-00-00 00:00:00	time that all all updates were last completed	0000-00-00 00:00:00		

## text\_organism indices

Keyname	Type	Field
Id	PRIMARY	Id

**text\_englishdict table**

Last updated February 16, 2004

<b>Database:</b>	seqhound
<b>Table:</b>	text_englishdict
<b>Module:</b>	text
<b>Definition:</b>	This table holds an English Dictionary from Moby project <a href="http://www.dcs.shef.ac.uk/research/ilash/Moby/"><u>http://www.dcs.shef.ac.uk/research/ilash/Moby/</u></a>
<b>Program:</b>	
<b>Organization:</b>	Oxford University
<b>File:</b>	

## text\_englishdict table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(11)	No		Auto incremented identifier. Aunique identifier for this word-pos combination.	1		
word	char(16)	Yes	NULL	the word	the		
pos	char(10)	Yes	NULL	part of speech	Det		
freq	int(11)	Yes	NULL	frequency in the bnc per million words	61847		
count	int(11)	Yes	NULL	how many bioname identifiers does this word refer to	1		
source	char(10)	Yes	NULL	pubmed = this word is in th pubmed stopword list stop = present in other stopword lists other possibilities	pubmed		

## text\_englishdict indices

Keyname	Type	Field
Id	PRIMARY	Id
Word	Index	Word
Pos	Index	Pos
Freq	Index	Freq
Count	Index	count





**text\_bncorpus table**

Last updated February 16, 2004

<b>Database:</b>	seqhound
<b>Table:</b>	text_bncorpus
<b>Module:</b>	text
<b>Definition:</b>	This table holds the British National Corpus.
<b>Parser:</b>	
<b>Organization:</b>	Oxford University ( <a href="http://www.natcorp.ox.ac.uk/">http://www.natcorp.ox.ac.uk/</a> )
<b>File:</b>	<a href="ftp://ftp.itri.bton.ac.uk/bnc/"><u>ftp://ftp.itri.bton.ac.uk/bnc/</u></a>
<b>Note:</b>	this file is not distributed with SeqHound

## text\_bncorpus table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		auto incremented identifier. A unique identifier for this word.	1		
word	char(160)	No		the word	the		
freq	int(11)	No	0	frequency in the corpus	6187267		
pos	char(16)	No		part of speech	at0		
files	int(11)	No	0	number of files (out of 4120) that this word appears in	4120		

## text\_bncorpus indices

Keyname	Type	Field
Id	Primary	Id
Word	Index	Word
Freq	Index	Freq

**text\_pattern table**

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_pattern

**Module:** text

**Definition:** This table holds a list of regular expressions used to detect mentions of biophysical interactions between two given names that appear in a sentence.

**Program:**

**Source Org:** Blueprint

**Source File:**

**Note:**

## text\_pattern table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	varchar(8)	No		a unique identifier for this regular expression.	9920		
parentid	varchar(8)	Yes	NULL	the parent identifier if this expression is derived from another.	99		
class	int(11)	No	0	does this expression identify an interaction (1) or the absence of an interaction (-1).	1		
score	double	Yes	NULL	score for this regex	0.85		
regex	text	Yes	NULL	the regular expression	A(\S*\s+)\{0,4\}\S*B(\S*\s+)\{0,4\}\S*heterodimer		

## text\_pattern indices

Keyname	Type	Field
Id	Primary	Id
Parentid	Index	Parentid

**text\_stopword table**

Last updated February 16, 2004

**Database:** seqhound

**Table:** text\_stopword

**Module:** text

**Definition:** This table contains a complete list of stopwords.

1. <http://www.ncbi.nlm.nih.gov/entrez/query/static/help/pmhelp.html#Stopwords>

2. <ftp://ftp.cs.cornell.edu/pub/smart/english.stop>

3. Manually collected from bioname table

**Program:**

**Source Org:** Various (see definition)

**Source file:**

**Note:**

## text\_stopword table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No		Auto incremented identifier	1		
word	char(16)	No		the stop word	a		
source	char(16)	Yes		Source of this stop word	PubMed help		

## text\_indices

Keyname	Type	Field
Id	Primary	Id
Word	Index	word

**Gene Ontology Association (GOA) module**

Last updated March 21, 2005

**purpose:**

The GOA Module provides Gene Ontology (GO) information for all possible organisms that have GO terms. GO terms are controlled vocabulary for molecular function, biological process and cellular component of gene products. GO Annotation assignments are derived from <ftp://ftp.geneontology.org/pub/go/gene-associations/> and from <ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/gene2go.gz>.

The GO terms are linked to identifiers provided by the curating database (e.g. FlyBase) AND to the NCBI Gene Info identifier (GI) using the SeqHound DBXref Module.

The GOA Module contains

- GO terms
- Literature references like PubMed IDs
- Evidence codes between the gene product and the GO term
- Object types that get annotated such as gene, transcript, protein or protein structure
- Gene symbols or other associated text
- Taxonomic identifiers
- Date on which the annotation was made.

**module:**

GOA

**input files:**

All files found under gene ontology FTP site

<ftp://ftp.geneontology.org/pub/go/gene-associations/>

NCBI EntrezGene



<ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/gene2go.gz>

**tables altered:**

goa\_association

goa\_gigo

goa\_reference

goa\_seq\_dbxref

goa\_with

goa\_xdb

**parsers:**

goa\_parser\_cluster.pl

goa\_geneparser\_cluster.pl

seqhound\_gigo\_cluster.pl

seqhound\_gigoPDB\_cluster.pl

seqhound\_gigoCGEN\_cluster.pl

source code location:

*The parsers for this module have been updated to work in a cluster configuration. Source code is unavailable at the time of this manual release but will be released with the next code release.*

**config file dependencies:**

*seqhound@blueprint.org*

Version 3.2

The relevant configuration file is:

*dbxref.ini* (for Unix platforms)

**command line parameters:**

NA

**associated scripts:**

The following shell scripts execute the parsers.

NA

**error and run-time logs:**

*dbxref\_errors.log*

*dbxref\_updatecron.log*

*goa\_parser.log*

*dbxref\_parser.log*

**troubleshooting:**

**additional info:**

The Gene Ontology Consortium documentation is located at

*<http://www.geneontology.org/GO.contents.doc.html>*

*[ftp://ftp.geneontology.org/pub/go/doc/GO.xrf\\_abbs\\_spec](ftp://ftp.geneontology.org/pub/go/doc/GO.xrf_abbs_spec)*

A diagram showing table relationships is available at the end of GOA module. The tables also appear in [http://www.blueprint.org/seqhound/api\\_help/docs/SeqHound\\_Schema\\_Prod.pdf](http://www.blueprint.org/seqhound/api_help/docs/SeqHound_Schema_Prod.pdf). with datatypes but without relationships.

**goa\_seq\_dbxref table**

Last updated March 23, 2005

<b>Database:</b>	Seqhound
<b>Table:</b>	goa_seq_dbxref
<b>Module:</b>	GOA
<b>Definition:</b>	The main table of the GOA Module. This table contains entries for a database record ID (e.g.: Fly Base ID FBgn0013277) and the information associated with the record that can be found in gene_association files from Gene Ontology. The record is the base to obtain other information like GO terms, reference and other annotation.
<b>Source org:</b>	Gene Ontology <i>ftp://ftp.geneontology.org/pub/go/gene-associations/</i>
<b>Source file:</b>	See module description above.
<b>Parser:</b>	See module description above.

## goa\_seq\_dbxref table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(10)	No	Auto incremented integer.	Internal primary record identifier.	2099131	Assigned by source document parser.	NA
goa_xdb_id	int(10)	No	0	Database identifier from table goa_xdb. For example, an xdb_id of 4 indicates the Candida Genome Database.	5	See note below.	NA
xref_key	varchar(30)	Yes	NULL	Record ID (in the above-mentioned database) which points to the object being annotated.	PrID1098818	See note below.	NA
taxid_1	int(11)	No	0	Taxonomic identifier of the species encoding the gene product.	11676	See note below.	NA
taxid_2	int(11)	Yes	0	Taxonomic identifier of the species where the gene product acts (in the manner described by the GO annotation). For example, if the protein is from a virus, the host organism might be listed here.	0	See note below.	NA
type_id	varchar(20)	No		Indicates the type of object being annotated (gene, transcript, protein etc.)	protein	See note below.	NA

symbol	varchar(30)	No		A unique and valid symbol to which the xref_key is matched. It can be an ORF name, gene product symbol or any other identifier.	GI424263	See note below.	NA
full_name	varchar(255)	Yes	NULL	Name of gene or gene product.	cell surface glycoprotein gp138	See note below.	
synonym	varchar(50)	Yes	NULL	Gene symbol or other text.	fusA	See note below.	
timestamp	timestamp(14)	Yes	NULL	When was this entry last updated.	20050317004300		

Note: columns in this table may be mapped to the GO Annotation flat file format described at <http://www.geneontology.org/GO.annotation.shtml#file>. See the Figure entitled “GOA relationships” below.

goa\_seq\_dbxref indices

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	2138717	id
goa_xdb_id_idx	INDEX	16	goa_xdb_id
xref_key_idx	INDEX	2138717	xref_key
symbol_idx	INDEX	2138717	
igoa_seq_dbxref_ts	INDEX	16	timestamp

**goa\_association table**

Last updated March 21, 2005

<b>Database:</b>	Seqhound
<b>Table:</b>	goa_association
<b>Module:</b>	GOA
<b>Definition:</b>	This table contains GO term information associated with the Record ID (goa_seq_xref.xref_key).
<b>Source file:</b>	See module description above.
<b>Parser:</b>	See module description above.



## goa\_association table

Field	Type	Null	Default	Column Definition	Example	Source	API
id	int(10)	No	Auto increment integer.	Internal unique identifier for this record.	33	This identifier is incremented by the source file parser.	NA
is_not	char(1)	Yes	F	Flags that modify the interpretation of an annotation. A GO ID with a Not in this field means that a particular gene product is NOT associated with a particular GO term.  For more information please read "Using the Qualifier column" from <a href="http://www.geneontology.org/GO.annotation.html">http://www.geneontology.org/GO.annotation.html</a> "T" indicates that "NOT was found in this column." "F" indicates that "NOT" was not found.	F	See note below.	NA
go_id	int(10)	No	0	Gene Ontology identifier for the term attributed to the object described by goa_seq_xref.xref_key.	3677	See note below.	NA
goa_seq_dbxref_id	int(10)	No	0	Foreign key pointing to goa_seq_dbxref.id	7	See note below.	NA
assigned_by	int(10)	No	0	database that made the annotation.	117	See note below.	
code	char(4)	No		Evidence Code. One of IMP, IGI, IPI, ISS, IDA, IEP, IEA, TAS, NAS, ND, IC.	IEA	See note below.	NA
date	char(8)	Yes	NULL	Date on which the annotation was made.	20040107	See note below.	NA
timestamp	timestamp	Yes	NULL	Date when this record was last modified.	20050316093600	See note below.	

Note: columns in this table may be mapped to the GO Annotation flat file format described at <http://www.geneontology.org/GO.annotation.shtml#file>. See the Figure entitled "GOA relationships" below.

## goa\_association indices

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	8720064	id
go_id	UNIQUE	8720064	go_id
			goa_seq_dbxref_id
			code
goa_id_idx	INDEX	21268	go_id
goa_seq_dbxref_id_idx	INDEX	8720064	goa_seq_dbxref_id
	INDEX	19	assigned_by
igoa_association_ts	INDEX	19	

**goa\_reference table**

Last updated March 21, 2005

<b>Database:</b>	Seqhound
<b>Table:</b>	goa_reference
<b>Module:</b>	GOA
<b>Definition:</b>	<p>This table contains the reference identifier for the GO annotation. Reference identifiers are unique identifiers appropriate to a database authority for the attribution of the go_id to the Record ID. They may be a literature reference or a database record.</p>
<b>Source file:</b>	See module description above.
<b>Parser:</b>	See module description above.

## goa\_reference table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(10)	No	Auto increment integer.	A unique identifier for this record.	1	This value is autoincremented by the source file parser.	NA
goa_association_id	int(10)	No	0	Foreign key pointing to goa_association.id.	530508	See note below.	NA
goa_xdb_id	int(10)	No	0	Database identifier that made the reference. Foreign key pointing to goa_xdb.id..	49	See note below.	NA
xref_key	varchar(20)	No		Reference Identifier. For example, if the reference is a published paper that has a PubMed ID, the PubMed ID number will be in this field	MGI:1354194	See note below.	NA
timestamp	timestamp(14)	Yes	NULL	When was this entry last updated?	20050316203235		NA

Note: columns in this table may be mapped to the GO Annotation flat file format described at <http://www.geneontology.org/GO.annotation.shtml#file>. See the Figure entitled “GOA relationships” below.

## goa\_reference indices

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY		id
goa_association_id_idx	INDEX	8424423	goa_association_id
goa_xdb_id_idx	INDEX	18	goa_xdb_id
igoa_reference_ts	INDEX	17122	timestamp
igoa_ref_key		2864	xref_key



**goa\_with table**

Last updated March 21, 2005

<b>Database:</b>	Seqhound
<b>Table:</b>	goa_with
<b>Module:</b>	GOA
<b>Definition:</b>	This table is used to hold additional identifiers for annotations using certain evidence codes. For more information please see "With (or) From" section at <a href="http://www.geneontology.org/GO.annotation.html">http://www.geneontology.org/GO.annotation.html</a>
<b>Source files:</b>	See module description above.
<b>Parsers:</b>	See module description above.

## goa\_with table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(10)	No	Auto increment integer.	A unique identifier for this record.	1	This column is autoincremented by the source file parser.	NA
goa_association_id	int(10)	No	0	Foreign key pointing to goa_association.id	530509	See note below.	NA
goa_xdb_id	int(10)	No	0	Database Identifier that made the 'with' annotation.	37	See note below.	NA
xref_key	varchar(20)	No		Reference identifier.	IPR001601	See note below.	NA
key_type	int(10)	No	0	Type of the symbol annotated: Type of the symbol annotated: 1=gene_symbol, 2=allele_symbol, 3=gene_id, 4=sequence_id, 5=go_id.	0	See note below.	NA
goa_seq_dbxref_id	int(10)	No	0	Foreign key pointing to goa_seq_dbxref.id for quick look-up from a 'with' annotation back to the record id.	0	See note below. This field is not currently implemented.	NA
timestamp	timestamp(14)	Yes	NULL	When was this entry last updated?	20050317005834		NA

Note: columns in this table may be mapped to the GO Annotation flat file format described at <http://www.geneontology.org/GO.annotation.shtml#file>. See the Figure entitled "GOA relationships" below.

**goa\_with indices**

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	1234449	id
goa_association_id_idx	INDEX	1234449	goa_association_id
goa_xdb_id_idx	INDEX		goa_xdb_id
goa_seq_dbxref_id_idx	INDEX	17	goa_seq_dbxref_id
igoa_with_ts	INDEX	1239	timestamp
igoa_with_ref	INDEX	94957	xref_key



**goa\_xdb table**

Last updated March 21, 2005

**Database:** Seqhound

**Table:** goa\_xdb

**Module:** GOA

**Definition:** The table goa\_xdb contains metadata about the organizations which contribute to the GO. There is a one to one relationship between abbreviations and URLs where data can be retrieved. A single URL which can be queried using database ids is referred to as a datasource. Each organization may have multiple datasources. Each abbreviation identifies one section of the file which provides the abbreviation and full name of that data source, the object type which is retrieved, an example database id, the generic url which identifies that data source uniquely and globally and the syntax of an actual query request with parameters filled in.

For more information please read:

*[ftp://ftp.geneontology.org/pub/go/doc/GO.xrf\\_abbs\\_spec](ftp://ftp.geneontology.org/pub/go/doc/GO.xrf_abbs_spec)*

**Source files:** See module description above.

**Parsers:** See module description above.

## goa\_xdb table

Field	Type	Null	Default	Column_Definition	Example	Source	API
id	int(11)	No	Auto increment integer.	A unique identifier for this record. Note that this identifier is unstable and may change from one release to another.	76	This column is autoincremented by the source file parser.	NA
abbreviation	varchar(50)	Yes	NULL	Database abbreviation.	SWISS-PROT	See note below.	NA
name	varchar(255)	Yes	NULL	Database name or description.	Swiss-Prot protein database.	See note below.	NA
object	varchar(255)	Yes	NULL	type of identifier returned from the data source: Accession number Locus identifier call number Gene symbol etc.		See note below.	NA
example	varchar(50)	Yes	NULL	An example database identifier.	Swiss-Prot:P45867	See note below.	NA
generic_url	varchar(255)	Yes	NULL	The root or representative URL for this data source.	<a href="http://ca.expasy.org/sprot/">http://ca.expasy.org/sprot/</a>	See note below.	NA
url_syntax	varchar(255)	Yes	NULL	A string to which one can append a database ID and get a valid URL query for the object referenced by that id. There is no wild card to represent the database ID, it is simply appended to the end of the string.	<a href="http://www.expasy.ch/cgi-bin/sprot-search-ac?">http://www.expasy.ch/cgi-bin/sprot-search-ac?</a>	See note below.	NA
url_example	varchar(255)	Yes	NULL	An example of what the url_syntax will look like	<a href="http://www.expasy.ch/cgi-bin/sprot-search-ac?P45867">http://www.expasy.ch/cgi-bin/sprot-search-ac?P45867</a>	See note below.	NA
timestamp	timestamp	Yes	NULL	When was this entry added to the database.	20050317012015	See note below.	

Note: columns in this table may be mapped to the GO Annotation flat file format described at <http://www.geneontology.org/GO.annotation.shtml#file>. See the Figure entitled “GOA relationships” below.

**goa\_xdb indices**

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	58	id
igoa_xdb_ts	INDEX	2	timestamp

**goa\_gigo table**

Last updated March 21, 2005

**Database:** Seqhound

**Table:** goa\_gigo

**Module:** GOA

**Definition:** Pre-computed list of gi-go pairs.

Since SeqHound database is NCBI GI Centric and GOA is organism database ID centric; the process to get a GO term given a GI can be complicated and time consuming, for this reason a pair list of gi-go were pre-computed and stored in goa\_gigo table. The SeqHound core database (mainly accdb and redund tables) is required to run these parsers.

**Source files:** See module description above.

**Parsers:** See module description above.

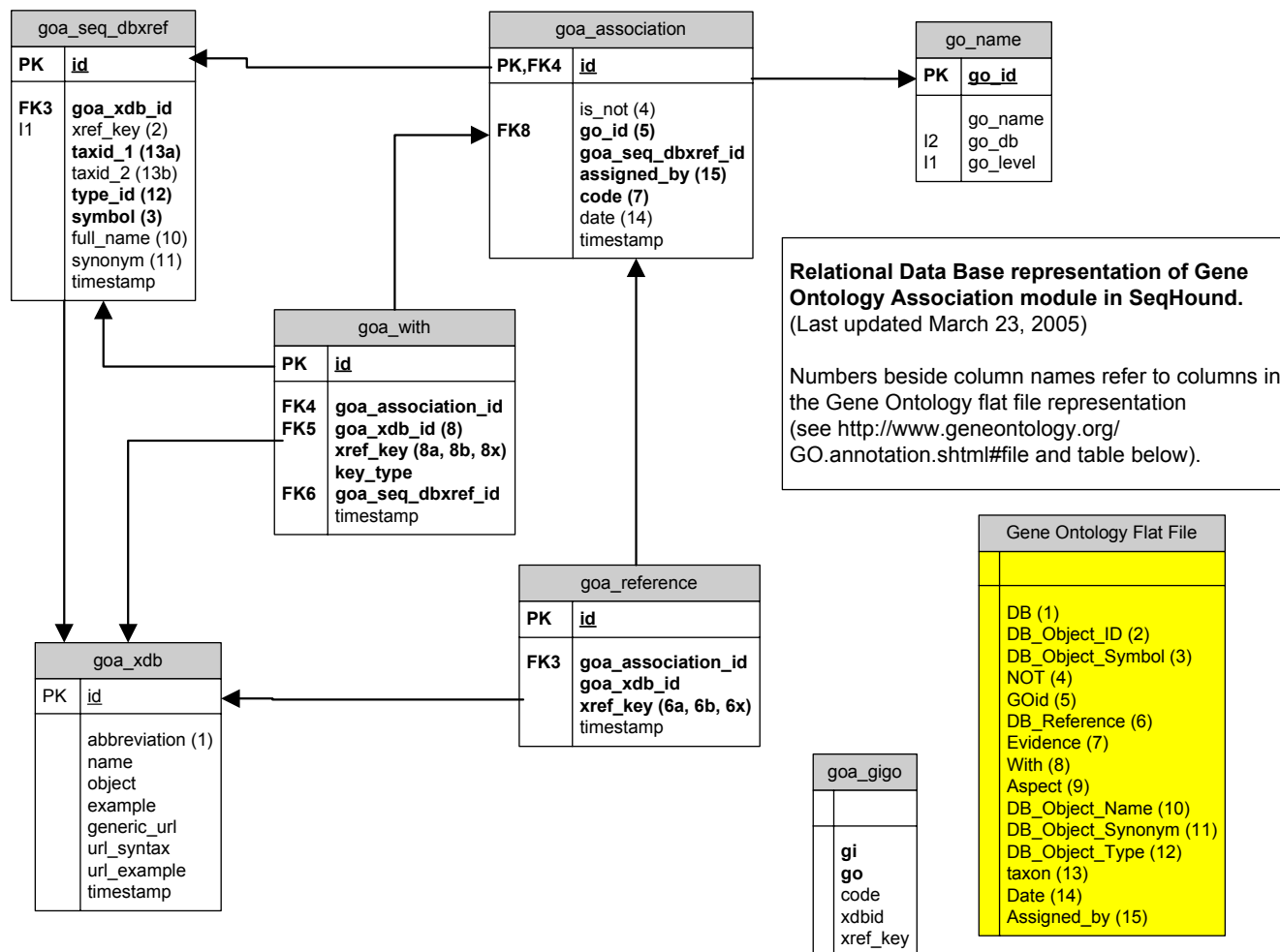
## goa\_gigo table

Field	Type	Null	Default	Column_Definition	Example	Source	API
gi	int(11)	No	0	Gene Info Identifier for an NCBI sequence record	6552303	The record indicated by the last two columns of this table are converted to a matching Gene Info Identifier using the DBXref module.	NA
go	int(11)	No	0	Gene Ontology Identifier.	6281	From table goa_association:gi_id	NA
code	varchar(4)	Yes	NULL	Evidence Code. One of IMP, IGI, IPI, ISS, IDA, IEP, IEA, TAS, NAS, ND, IC.  For more information please read <a href="http://www.geneontology.org/GO.evidence.html">http://www.geneontology.org/GO.evidence.html</a>	TAS	From table goa_association:code	NA
xdb_id	int(11)	Yes	NULL	Identifier for database. See also xref_key below. From table goa_xdb.	103	From table goa_xdb.id	NA
xref_key	varchar(30)	Yes	NULL	Identifier in database (see previous column) pointing to object that was originally annotated.	P38398	From table goa_seq_dbxref:xref_key.	NA
timestamp	timestamp(14)	Yes	NULL	When was this entry last updated?	20050316192314		NA

**goa\_gigo indices**

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	7936320	gi
			go
gi_idx		3968160	gi
	INDEX		go
xdb_id_idx	INDEX	22	xdb_id
xref_key_idx	INDEX	7936320	xref_key
igoa_gigo_ts	INDEX	22	timestamp

## Gene Ontology Module Diagram







**dbxref module**

Last updated March 22, 2005

**purpose:**

The purpose of the SeqHound dbxref module is to have a centralized data source (cross references) where related information can be found from a given ID. By using the dbxref module, it is possible to find one to "n" relationships between IDs from 3rd party databases for DNA, Protein Sequences, Domains and Interactions, GenBank Accession Numbers, Swiss-Prot, LocusLink, SGD, MGD, ZFIN, FB, PFAM, SMART, etc. See the "Explanation of data table structure" below.

**Explanation of data table structure:**

The dbxref table is the core of the dbxref module. This explanation refers to that table.

Dbxref represents cross references between 3<sup>rd</sup> party databases and GeneBank. This table is created by parsing 3<sup>rd</sup>. party flat files and creating records in "dbxref" for each record parsed. dbxref is a self-referencing table.

The value in the "record\_id" field may represent one of two things:

1. A source record:

If the value of "parent\_id" in the row is zero (0) it is referred to as an object id. Its "record\_id" is the identifier/primary id for a record in the database ("source\_db") from which cross-references have been retrieved.

For example, from the dbxref table content example listed below.

**P38903** is a Primary ID in SwissProt Database

**S00055403** is a Primary ID in SGD Database.

2. A database cross-reference found in a source record. The retrieved cross-references are stored as a combination of
  - a) Source database ("source\_db").
  - b) an identifier for a record in that database ("record\_id").
  - c) the record that the cross-reference was found in. The field "parent\_id" contains an integer. This integer refers to the row in this table (see "id") that contains the identifier of the record from which this cross reference was retrieved.
  - d) cross references are retrieved from some field in a record. The name of this field is recorded in the "field" column or if there are no field names then the column number is recorded. For example : Col11 or Col14.

### Examples:

id	source_db	record_id	parent_id	link	field	cv
1	SP	P38903	0	0	ID	0
2	GB	U06630	1	0	DR	1
3	GB	AAB38372	1	2	DR	2
4	GB	S79635	1	0	DR	1
5	GB	AAB35312	1	4	DR	2
6	GB	X87331	1	0	DR	1
7	GB	CAA60763	1	6	DR	2
8	GB	Z74922	1	0	DR	1
9	GB	CAA99203	1	8	DR	2
10	InterPro	IPR001757	1	0	DR	3
11	Pfam	PF00689	1	0	DR	3
12	SGD	S0005540	1	0	DR	0
13	SP	P47096	0	0	ID	0
14	GB	Z49525	13	0	DR	1
15	GB	CAA89550	13	14	DR	2
16	GB	X87297	13	0	DR	1
17	IPR	IPR007113	13	0	DR	0
18	SGD	S00055403	0	0	Col14	0
19	GB	1420113	18	0	GI	0
20	GermOnline	143602	18	0	Col11	0
21	DIP	4191	18	0	Col11	4
22	GB	6324588	18	0	GI	0

23	GB	AX596518	18	0	Col1	1
24	CandidaDB	CA1247	18	0	Col1	0
25	GB	CAA99203	18	0	Col1	2
26	InterPro	IPR002554	18	0	Col1	0
27	GB	NP_014656	18	0	Col1	2
28	SP	P38903	18	0	Col1	0
29	PIR	S54620	18	0	Col1	0
30	GB	S79635	18	0	Col1	1
31	GB	U06630	18	0	Col1	1
32	GB	X87331	18	0	Col1	1
33	MIPS	YOR014W	18	0	Col1	0
34	GB	Z74922	18	0	Col1	1
+-----+-----+-----+-----+-----+-----+-----+						

The "link", "field" and "cv" fields in the dbxref table help specify where the information is coming from. These are described below.

### *link*

The link field was created to support some databases (for example Swiss-Prot) which may store more than one cross-reference in one field. For example a swiss-prot record for a protein may contain a "DR" field that lists two EMBL identifiers. ie.:

```
DR    EMBL; U06630; AAB38372.1; -.
```

The first identifier is a cross-reference for a nucleotide record in EMBL that encodes a protein (second identifier).

The link field was created to capture this relationship between the two cross-references. The integer in the link field points to a row in the dbxref table (see "id") and indicates that the current cross-reference is linked (or comes after) some other cross reference in the same source record and field.

The exact meaning of the database cross-reference can be discerned from the "cv" (Controlled Vocabulary) column of this table. In this example, one cross-reference would be labeled as "nucleotide" and one would be labeled as "protein".

2: points to record "id=2" meaning that protein sequence identifier "AAB38372" comes after the identifier "U06630" in the DR field.

0 : if no relationship exists or if the dbxref is the first one in a list

*field*

This describes the field in the source record where the database cross-reference was found. For example, “DR” is a field name in Swiss-Prot records indicating a Database Reference. Alternatively, a column number might be listed here if the source file was a tab-delimited text file.

*cv*

cv is a controlled vocabulary term that is used to describe the type of record that the database cross-reference is pointing to. This controlled vocabulary is simple at the moment and may be expanded in future. Briefly,

- 1 indicates a DNA sequence record
- 2 indicates a protein sequence record
- 3 RNA
- 100 Swiss-Prot record
- 101 Trembl record
- 110 Swiss-Prot secondary accession
- 111 Trembl secondary accession
- 0 means not defined.

**source code location:** /siri/seqhound/dbxref

**input Files:** See summary table.

**parsers:** See summary table.

**config file dependencies:**

dbxref.ini

**command line parameters:** See summary table.

**example use:** See summary table.

**associated scripts:**

The following shell scripts execute the parsers.

*dbxref\_cron\_monthly.sh*

*dbxref\_cron.sh*

**error and run-time logs:**

*dbxref\_errors.log*

*dbxref\_updatecron.log*

*dbxref\_parser.log*

Error messages are sent by email when *dbxref\_cron\_monthly.sh* is run

troubleshooting:

Check the email sent by *dbxref\_cron\_monthly.sh* to find out if any parsers had problems. For more details consult the log files.

**additional info:**

**Table summarizing input files, parsers and command line parameters for dbxref module.**

<b>Input file</b>	<b>parser</b>	<b>command line parameters</b>
<i>ftp://expasy.org/databases/uniprot/knowledgebase/uniprot_sprot.dat.gz</i>	dbxref_sptr_cluster.pl	<b>XREF_SP</b>
<i>ftp://expasy.org/databases/uniprot/knowledgebase/uniprot_trembl.dat.gz</i>	dbxref_sptr_cluster.pl	<b>XREF_TREMBL</b>
<i>http://flybase.bio.indiana.edu/allied-data/extdb/external-databases.txt</i>	dbxref_fb_cluster.pl	none
<i>ftp://ftp.sanger.ac.uk/pub/databases/wormpep/wormpep.table</i>	dbxref_wb_cluster.pl	none
<i>ftp://ftp.informatics.jax.org/pub/reports/MRK_Sequence.rpt</i>	dbxref_mgi_cluster.pl	none
<i>ftp://ftp.informatics.jax.org/pub/reports/MRK_SwissProt_TrEMBL.rpt</i>	dbxref_mgi_cluster.pl	none
<i>ftp://genome- ftp.stanford.edu/pub/yeast/data_download/chromosomal_feature/dbxref.tab</i>	dbxref_sgd_cluster.pl	none
<i>ftp://ftp.afcs.org/pub/mpdata/afcsflat.txt</i>	dbxref_AFCS_cluster.pl	<b>AFCS</b>
<i>ftp://ftp.tigr.org/pub/data/a_thaliana/ath1/DATA_RELEASE_SUPPLEMENT/release_5.genbank_accessions.txt.gz</i>	dbxref_tigr_ath_cluster.pl	none
<i>ftp://ftp.geneontology.org/pub/go/gp2protein/gp2protein.tigr_ath</i>	dbxref_tigr_ath_cluster.pl	none
<i>ftp://ftp.blueprint.org/pub/SeqHound/Private/DDB/dictybaseid_gb_accession.txt.gz</i>	dbxref_ddb_cluster.pl	none
<i>ftp://rgd.mcw.edu/pub/data_release/genbank_to_gene_ids.txt</i>	dbxref_rgd_cluster.pl	none
<i>http://zfin.org/data_transfer/Downloads/genbank.txt</i>	dbxref_zfin_cluster.pl	none
<i>http://zfin.org/data_transfer/Downloads/refseq.txt</i>	dbxref_zfin_cluster.pl	none
<i>ftp://ftp.geneontology.org/pub/go/gp2protein/gp2protein.zfin</i>	dbxref_zfin_cluster.pl	none
<i>ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/HUMAN/human.xrefs.gz</i>	dbxref_goa_xrefs_cluster.pl	none

<b>Input file</b>	<b>parser</b>	<b>command line parameters</b>
<i>ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/MOUSE/mouse.xrefs.gz</i>	dbxref_goa_xrefs_cluster.pl	none
<i>ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/RAT/rat.xrefs.gz</i>	dbxref_goa_xrefs_cluster.pl	none
<i>ftp://ftp.sanger.ac.uk/pub/yeast/pombe/Mappings/gp2swiss.txt</i>	dbxref_DBs_SPTR_cluster.pl	<b>-o spombe</b>
<i>ftp://ftp.geneontology.org/pub/go/gp2protein/gp2protein.tair</i>	dbxref_DBs_SPTR_cluster.pl	<b>-o tair</b>
<i>ftp://ftp.geneontology.org/pub/go/gp2protein/gp2protein.tigr_cmr</i>	dbxref_DBs_SPTR_cluster.pl	<b>-o tigr_cmr</b>
<i>ftp://ftp.geneontology.org/pub/go/gp2protein/gp2protein.unigene</i>	dbxref_DBs_SPTR_cluster.pl	<b>-o unigene</b>
<i>ftp://ftp.geneontology.org/pub/go/gp2protein/gp2protein.vida</i>	dbxref_DBs_SPTR_cluster.pl	<b>-o vida</b>
<i>ftp://ftp.ncbi.nih.gov/gene/DATA/gene2accessions.gz</i>	dbxref_gene_cluster.pl	
<i>ftp://ftp.ncbi.nih.gov/gene/DATA/mim2gene.gz</i>	dbxref_gene_extra.pl	<b>-d OMIM</b>
<i>ftp://ftp.ncbi.nih.gov/gene/DATA/gene2unigene.gz</i>	dbxref_gene_extra.pl	<b>-d UNIGENE</b>

**dbxref table**

Last updated March 22, 2005

**Database:** SeqHound

**Table:** dbxref

**Module:** dbxref

**Definition:** The dbxref table is the core of the dbxref module.  
It represents cross references between 3rd party databases and GeneBank. This table is created by parsing 3rd. party flat files and creating records in "dbxref" for each record parsed. dbxref is a self-referencing table. See "Explanation of data table structure" above.

**Source org:** multiple - [see summary table](#)

**Source file:** multiple - [see summary table](#)

**FTP script:** multiple - [see summary table](#)

**Parser:** multiple - [see summary table](#)



## dbxref table

Field	Type	Null	Default	Column_Definition	Example	Source <sup>1</sup>	API
id	int(11)	No	Auto increment	Primary key identifier for this entry.	2	All SQL insert statements will auto increment this field.	NA
source_id	int(11)	No	0	Foreign key pointing to an identifier for a database, in table dbxrefsourcecdb. For example, 1 = Swiss-Prot, 2 = GenBank...etc.	2	See dbxrefsourcecdb:source_id.	NA
record_id	char(30)	No		Record identifier in database mentioned in previous column. The format of this identifier will be that of the source database.	AAD12597		NA
parent_id	int(11)	No	0	Link pointing to "id" (first column in this table) to the record that is the source id of the Cross-Reference. An entry of "0" indicates that this is a source record from which database cross-references are parsed.	1	See dbxref:id	NA
link	int(11)	Yes	NULL	The link field was created to support some databases (for example Swiss-Prot) which may store more than one cross-reference in one field. See the section above "Explanation of data table structure".	0	SQL insert statements for proteins that have dbXref to the nucleotide will have the "link" field pointing to the "id" record of the nucleotide.  If this relationship cannot be established the "link" value will be set to 0.	NA
field	char(20)	Yes	NULL	Describes the field in the source record where the database cross-reference was found. For example, "DR" is a field name for Database Reference found in Swiss-Prot records.	Col6		NA

cv	int(11)	Yes	NULL	Controlled vocabulary. This will be used to describe the type of record that the cross-reference is pointing to. See the section above “Explanation of data table structure”.	2		NA
timestamp	timestamp(14)	Yes	NULL	When was this entry last updated?	0	20050317201847	NA

1. Multiple source files are parsed by multiple parsers. For details, *[see summary table](#)*.

### dbxref indices

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	16849002	id
i_source_id	INDEX	18	source_id
i_dbxref_id	INDEX	16849002	record_id
i_parent_id	INDEX	16849002	parent_id
i_link	INDEX		link
i_cv	INDEX	18	cv
idbxref_ts	INDEX	51843	timestamp

**dbxrefsourcedb table**

Last updated March 22, 2005

<b>Database:</b>	seqhound
<b>Table:</b>	dbxrefsourcedb
<b>Module:</b>	dbxref
<b>Definition:</b>	This table assigns an internal identifier to all data sources where cross-references are found. These identifiers are used in the source_id column of the dbxref table.
<b>Source org:</b>	multiple - <i><u>see summary table</u></i>
<b>Source file:</b>	multiple - <i><u>see summary table</u></i>
<b>FTP script:</b>	multiple - <i><u>see summary table</u></i>
<b>Parser:</b>	multiple - <i><u>see summary table</u></i>

dbxrefsourcedb table<sup>1</sup>

Field	Type	Null	Default	Column Definition	Example	Source <sup>1</sup>	API
source_id	int(11)	No	0	Database ID, primary key in table dbxrefsourcedb. For example, 1 = Swiss-Prot	1		NA
source_db	char(50)	No		Abbreviated name of database. Abbreviations for database names are the same as those used by the Gene Ontology group see: "GO.xrf_abbs" <i>ftp://ftp.geneontology.org/pub/go/doc/GO.xrf_abbs</i> This value is mandatory.	SP		NA
isprimary_db	tinyint(4)	No	0	Are cross-references retrieved from this database? 1 = YES 0 = NO	1		NA
timestamp	timestamp(14)	Yes	NULL	When was this entry added?	20050318070841		

1. The contents of this table are hand-edited as part of the dbxrefgoa.sql file. The entire contents of this table as of March 22, 2005 are listed below.

## dbxrefsourcedb indices

Keyname	Type	Cardinality	Field
PRIMARY	PRIMARY	33	source_id
i_source_id2	INDEX	33	source_id
i_source_db	INDEX	33	source_db
idbxrefsourcedb_ts	INDEX	2	timestamp



**Contents of dbxrefsourcedb table**

(last updated March 22, 2005).

source_id	source_db	isprimary_db	timestamp
1	SP	1	20050318070841
2	GB	0	20050318070841
3	PFAM	0	20050318070841
4	INTERPRO	0	20050318070841
5	MGI	1	20050318070841
6	SGD	1	20050318070841
7	SMART	0	20050318070841
8	ZFIN	1	20050318070841
9	FB	1	20050318070841
10	CG	0	20050318070841
11	TR	1	20050318070841
12	SPTR	1	20050318070841
13	WB	1	20050318070841
14	LL	0	20050318070841
15	CGEN	0	20050318070841
16	TIGR_ATH	1	20050318070841

17	REFSEQ	0	20050318070841
18	GENEDB_SPOMBE	1	20050318070841
19	DDB	1	20050318070841
20	GR	1	20050318070841
21	TAIR	1	20050318070841
22	TIGR_CMR	1	20050318070841
23	UNIGENE	1	20050318070841
24	VIDA	1	20050318070841
25	RGD	1	20050318070841
26	IPI	1	20050318070841
27	ENSEMBL	1	20050318070841
28	AFCS	1	20050318070841
29	HUGO	0	20050318070841
30	OMIM	0	20050318070841
31	PIR	0	20050318070841
32	GENE	1	20050318070841

## 6. Developing for SeqHound.

### Open source development.

SeqHound code is developed on a cvs tree internal to the Samuel Lunenfeld Research Institute by the members of the Blueprint Initiative Development team. The most stable current release is available at

*<ftp://ftp.blueprint.org/pub/SeqHound/>*

External developers are encouraged to discuss major additions or modifications to the system with the Project Manager at *[seqhound@blueprint.org](mailto:seqhound@blueprint.org)*.

Minor additions or corrections may also submitted to *[seqhound@blueprint.org](mailto:seqhound@blueprint.org)*.



## Code organization.

---

Note: The following section is under revision and will be updated shortly.

---

This document summarizes the contents in the directories under *slri/seqhound* and *slri/nblast* as of June of 2003. The contents of directories are first stated and then the purpose of the contents is given.

Under the current directory hierarchy, the seqhound directory contains the following directories (/) and files (\*):

*asn/*

*bioperl/*

*build/*

*cgi/*

*config/*

*db2/*

*domains/*

*examples/*

*genomes/*

*go/*

*html/*

*include/*

*include\_cxx*

*java/*

*lib/*

*locuslink/*

*parsers/*

*perl/*

*rps/*

*scripts/*

*shreadme/*

*shreadme\_cxx/*

*src/*

*src\_cxx/*

*taxon/*

*tindex/*

*updates/*

*yeast/*

*seqhound.mk\**

*seqhound\_cb.mk\**

*seqhound\_db2.mk\**

*seqhound\_odbc.mk\**

*seqhound\_rem.mk\**

*shreadme\**

*shreadme\_cxx\**

**asn/ :** contains the asn specifications (\*.asn) for various objects used in seqhound, auxiliary files used by datatool & asntool (\*.def), and the scripts used by asntool and datatool to autogenerate the objects in unix (\*.sh) and windows (\*.bat) platforms.

**bioperl/:** contains the files used in the SeqHound bioperl module

**build/:** contains a directory structure used to store the executables. As executables get compiled, they will be moved into their relevant directories inside the build directory. Executables using Codebase will be moved to *cb/*, db2 executables are moved to *db2/*, and odbc executables to *odbc/*. The appropriate directories will get created as the need arises.

**cgi/:** contains the source code for the cgi and web services.

**config/:** the configuration files used in seqhound remote & local applications.

**db2/:** scripts to create the tables in db2 and redund for db2.

**domains/:** source code for the domain module.

**examples/:** source code showing how to use some of the code in seqhound, ex the asn structures and functions, the C++ remote library.

**genomes/:** source code for the complete genome module.

**go/:** source code for the gene ontology module.

**html/:** documentation for SeqHound API, and some scripts for converting the documentation to html

**include/:** the \*.h files for SeqHound

**include\_cxx/:** the include files for the C++ remote library

**java/:** source code for the SeqHound Java remote library

**lib/:** libraries for SeqHound are copied here once compiled.

**locuslink/:** source code for the locuslink module.

<b>parsers/:</b>	source code for various parsers (mother, cbmmdb, cddb, redund, mmdbloc)
<b>perl/:</b>	the SeqHound perl module (deprecated in favor of bioperl)
<b>rps/:</b>	source code for the rps module (domname, rpsdb)
<b>scripts/:</b>	various scripts to retrieve flatfiles from NCBI, and to build SeqHound
<b>src/:</b>	source code for SeqHound (includes db layers for GO, HIST, LOCUSLINK, NBR, rpsdb, taxdb& core modules), the C remote API
<b>src_cxx/:</b>	source code for the remote C++ library.
<b>taxon/:</b>	source code for the taxon module parser.
<b>tindex/:</b>	source code for the text indexer.
<b>updates/:</b>	source code for the daily updates & histparser.
<b>yeast/:</b>	source code for importing yeast GO into Seqhound (never completed).
<b>*.mk:</b>	various auxiliary files used by SeqHound makefiles.
<b>shreadme*:</b>	readme files for C & C++ SeqHound

Under the current directory hierarchy, the *nblast* directory contains the following directories(/) and files(\*):

*asn/*

*db/*

*docs/*

*lib/*

*msvc/*

*scripts/*

*src/*

*nblast.mk\**

*nblastflags.mk\**

*asn/:* contains the ASN specifications for nblast.

*db/:* source code for nblast Codebase layer.

*docs/:* instructions for compiling nblast.

*lib/:* source code for the ASN nblast object.

*msvc/:* files for Microsoft Visual C++ project.

*scripts/* various scripts for retrieving NCBI Blast flatfiles & setting up NBlast.  
:  
*src/*: core source code for nbblast.  
*\*.mk*: auxiliary makefile for nbblast applications.

## **Adding/Modifying a remote API function to SeqHound.**

---

Note: This section is included for historical purposes and is being rewritten for a future release (5.0)

---

Overall steps:

1. open a new Bugzilla report
2. create the database search functionality
3. create the local API call
4. create the CGI call
5. add the remote calls (C/C++/Java/Bioperl/Perl)
6. test the new functionality in each of the layers
7. write/update documentation
8. inform technical writer
9. inform tester
10. update seqrem and local library in production server
11. In some rare cases, modifications may have to be made to the underlying data table structure, then SeqHound must be rebuilt in test and production environments before code is checked into cvs.
12. check in the new source code
13. update API website
14. close Bugzilla report

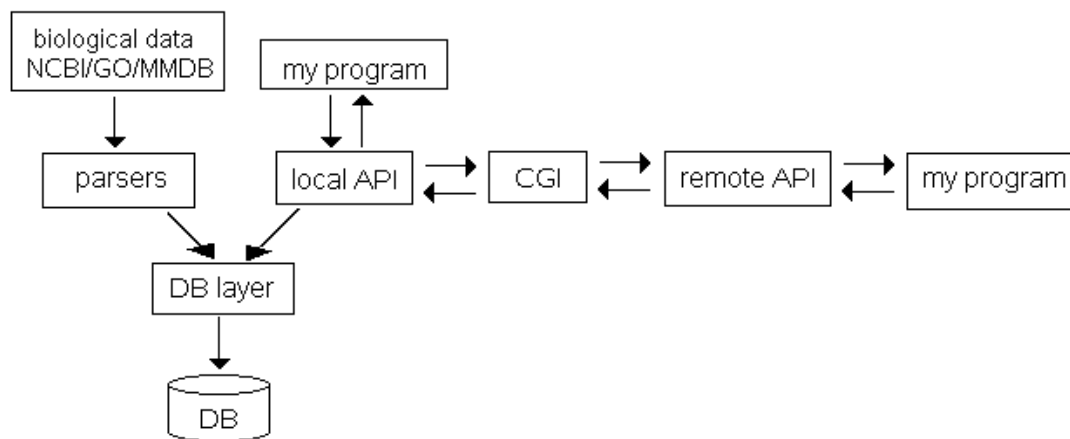
---

This document only goes over steps 2-5 in detail.

---

**Overall architecture of the SeqHound system.**

*The diagram shows how the various code layers in SeqHound.*



## 2. Create the Database Search Functionality

If adding a new data file, you will need to write the database search functions, one for each data file. Database Search functions are found in '*slri/seqhound/src/\*\_cb.c*' files.

'\*' is the name of the module that the searched table belongs to like 'GO' or 'intrez'. 'cb' refers to the database engine that the function talks to like CodeBase (cb) or ODBC compliant databases (odbc).

Functions that search data tables are typically called SearchXXX. New data files require you to define the ASN structure, typically each field in the data file corresponds to an ASN field.

Most of the existing databases already have search functions so you should not have to write these yourself. If they don't exist, then you should consult the instructions under Creating a new SeqHound module first. The details below are here to help you understand the database layer and to use Search functions in your API function.

```
Boolean LIBCALL SearchACCDB (StAccdbPtr PNTR ppac)
```

The search functions should find **ALL** instances of a key in the database and return them through the ASN pointer. Each ASN structure is a linked list, so multiple records can be retrieved. Pseudo code for the typical search function:

```
Boolean SearchXXX (ASNLinkList Pointer)
{
    foreach record that matches key in Pointer
        create a new ASNLinkList Node;
        fill node with record fields;
        join node to Pointer;
    end foreach

    return TRUE if key found
    else return FALSE;
}
```

As an example, the function

```
Boolean LIBCALL SearchACCDB (StAccdbPtr PNTR ppac)
```

finds records in the accdb table for an ODBC compliant database engine. This function is located in *slri/seqhound/src/intrez\_odbc.c*. The corresponding function for a CodeBase database engine has the same name but is found in *slri/seqhound/src/intrez\_cb.c*. The ASN structure that StAccdbPtr points to is defined in *slri/seqhound/asn/slristruc.asn* (search for StAccdb) and the corresponding C structure is defined in *slri/seqhound/include/objslristruc.h*. You need to know this when you use a database search function in your local API function.

## 3. Create The Local API Function

You will need to change one of three files in order to add new functionality to SeqHound.

- a) *go\_query.c* : to add functionality to gene ontology module API functions
- b) *ll\_query.c* : to add to locuslink module API functions
- c) *intrez.c* : to add new functionality to the other data tables.

Declare function prototype in *slri/seqhound/include/seqhound.h*. Try to group the functions logically in the *.c* and *.h* files.

The API functions follow a general logic. You should try to stick to that logic.  
Typical example of an API function:

```

Int4 LIBCALL SHoundFindAcc(CharPtr pcAcc)
{
    Int4 gi = 0;
    StAccdbPtr pac = NULL, pachead = NULL;
    Int2 res = 0;

    if ((pcAcc == NULL) || (strcmp(pcAcc, "n/a") == 0))
    {
        ErrPostEx(SEV_ERROR, 0, 0, "SHoundFindAcc: No accession.");
        return 0;
    }

    pac = StAccdbNew();
    pachead = pac;
    pac->access = StrSave(pcAcc);
    res = SearchACCDB(&pac);

    if (res == FALSE)
    {
        ErrPostEx(SEV_ERROR, 0, 0, "SHoundFindAcc: Search failed.");
        StAccdbFree(pachead);
        return 0;
    }
    else if (res == TRUE)
    {
        gi = pac->gi;
        pac = pac->next;
    }
    StAccdbFree(pachead);
    return gi;
}

```

Input integrity check

Each data table has specific asn structure.

Set the key in the asn structure database layer to search for the key. The asn structure will hold all the records that match the key.

Failed search. Free structure and return

Successful search - extract what you need and return it.

#### 4. Step 4. Create the CGI Function

Definitions:

##### CGI (Common Gateway Interface):

a program that allows 2 computers connected to the internet to communicate with each other.

##### HTTP (Hypertext Transfer Protocol):

a way of passing messages between the 2 computers. Is used for web services, eg CGI, servlets ... Format of HTTP:

[http://server\\_name/path/to/cgi/cgi\\_name?key1=value1&key2=value2&...](http://server_name/path/to/cgi/cgi_name?key1=value1&key2=value2&...)

##### Query string:

The portion of the HTTP call that is used to pass parameters to the CGI. Format of query strings: ?param1=value1&param2=value2....



## SeqHound's CGI

is `slri/seqhound/cgi/seqrem.c`.

### Remote users

can call `seqrem` on our servers using HTTP calls. Format of a HTTP call:

`http://seqhound.blueprint.org/cgi-bin/seqrem?fnct=SeqHoundFindAcc&acc=AA73235`

2 parts to edit in `seqrem.c`

1. `if` statement in function: `ProcessFnctRequest(CharPtr pfnc)`;
2. Add a corresponding CGI function call that calls the local API

---

There is currently an `if-else` statement in the `ProcessFnctRequest` function. It needs to be modified to check if the first field of the query string contains your new function name. If so, it will call a CGI function that calls your local API function.

---

```
SLRI_ERR ProcessFnctRequest(CharPtr pfnc)
{
    if(SeqHoundInit() == SLRI_FAIL) {
        MemFree(pfnc);
        return 1;
    } else if (strcmp(pfnc, "SeqHoundFindAcc") == 0) {
        if(SeqHoundFindAcc() == SLRI_FAIL) {
            MemFree(pfnc);
            return 1;
        }
        ...
    } else if(strcmp(pfnc, "YourNewFunction" ..
        if(SeqHoundYourNewFunction() ...
    }
}
```

Calling internal CGI function  
... see below

Your new function gets called here

Next you need to define a function with the prototype

```
SLRI_ERR SeqHoundYourNewFunction(void)
```

This function will extract the remaining keys in the query string portion of the HTTP call, and call the local API function you defined in step 3, then output the HTTP response to be sent back to the user.

`http://seqhound.blueprint.org/cgi-bin/seqrem?fnct=SeqHoundFindAcc&acc=AA73235`

```
SLRI_ERR SeqHoundFindAcc(void)
{
    Int4 gi = 0;
    Int4 IndexArgs = -1;

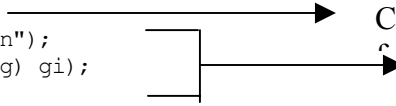
    printf("Content-type: text/html\r\n\r\n");
    if ((IndexArgs = WWWFindName(ginfo,"acc")) >= 0)
    {
        pcThis = WWWGetValueByIndex(ginfo,IndexArgs);
        pacc = StringSave(pcThis);
    }
    if ((pacc == NULL) || (strlen(pacc) == 0))
    {
        ErrPostEx(SEV_ERROR,0,0, " Failed to get parameter value.");
    }
}
```

Internal CGI function that calls the local API

Output the HTTP headers

Extract the remaining fields from the CGI

```
    fprintf(stdout, "SEQHOUND_ERROR Failed to get parameters.");  
    return SLRI_FAIL;  
}  
gi = SHoundFindAcc(pacc);  
fprintf(stdout, "SEQHOUND_OK\n");  
fprintf(stdout, "%ld\n", (long) gi);  
MemFree(pacc);  
return SLRI_SUCCESS;  
}
```



Calling the local API

Send output  
back to remote  
users using  
HTTP

## 5. Step 5. Create the Remote Calls

We have programs available in most of the widely used languages that allow SeqHound users to write programs in their favorite language, accessing SeqHound data without having to understand how everything works. Our remote programs in effect constructs the HTTP calls (described above), sends the HTTP calls to the server and then parses the server's return value and sends this back to the user's program.

Our remote interfaces are:

1. *slri/seqhound/src/seqhoundapi.c*
2. *slri/seqhound/src\_cxx/Seqhound.cpp*
3. *slri/seqhound/java/SeqHound.java*
4. *slri/seqhound/perl/SeqHound.pm*
5. *slri/seqhound/bioperl/SeqHound.pm*

```
Int4 LIBCALL SHoundFindAcc(CharPtr pcAcc)
{
```

```
    Char fpath[PATH_MAX];
    Int4 gi = 0;
```

```
    if(pcAcc == NULL)
```

```
    {
        ErrPostEx(SEV_ERROR,0,0, "Invalid parameter.");
        return 0;
    }
```

```
    sprintf(fpath,"%s?fnc=SeqHoundFindAcc&acc=%s", slri_cgipath, pcAcc);
```

```
    ErrPostEx(SEV_INFO,0,0, "SeqHoundFindAcc request: %s.\n", fpath);
```

```
    if(SHoundWWWGetfile(slri_sername, fpath) == 0)
```

```
    {
        ErrPostEx(SEV_ERROR,0,0, "SHoundWWWGetfile failed.");
        return 0;
    }
```

```
    gi = ReplyBSGetInteger();
```

```
    if (gi == 0)
```

```
    {
        ErrPostEx(SEV_INFO,0,0, "SeqHoundFindAcc returned zero.");
        return 0;
    }
```

```
    return gi;
}
```

Making the HTTP  
call.

Contains the server  
name path to CGI

Send the HTTP  
call

Get the return  
value.  
A family of  
ReplyBSGetXX  
X exists for

Similar logic is used in the remote C++, Java, and Perl libraries.

## Adding a new module to SeqHound

---

Note: This section is included for historical purposes and is being rewritten for a future release (5.0)

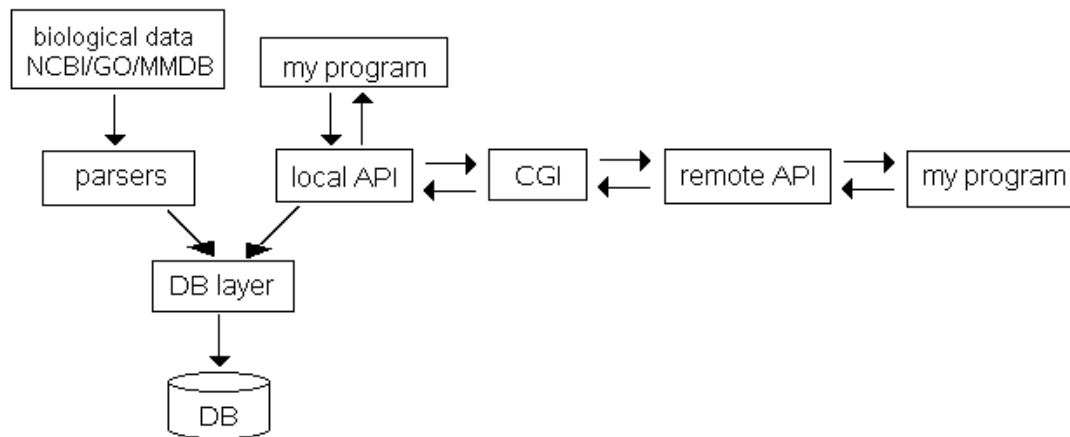
---

This section describes how to go about adding a new module to SeqHound from the developer's point of view. It is basically a description of the different files that have to be written for a new module and where they should go in the code tree. So this section may be used as a guide for looking at existing modules to find out what files are expected and where they are. However, historically, several programmers have added modules to SeqHound and have used different code organization schemes; therefore, historical modules may be organized differently. Going forward (as of November 2003) all modules will have the components that are organized in the way described below.

### 1. Start a new module project plan

Creating a new module begins with starting a new project plan (see Format for project reports). This should contain a clearly stated “*Need*” and “*Objective*” as well as a “*General approach*” to developing the module; this may include background information on the data resource being incorporated into SeqHound. The “*Detailed planning*” section will contain a section on “*Data table design*” and “*Code Organization*” that will be described further. See the DBXREF (Database cross-reference) module Project Plan as an example (*./slri/seqhound\_priv/dbxref/dbxref\_module\_desc.txt*).

The components of a new module are summarized in the diagram below. The rest of this section describes the creation of each of these components and their organization. File names in ***bold italics*** indicate files that may be looked at as examples.



## Database layer

### 2. Design the data table structure

The module's project description should contain a description of the final table design. See the example in the DBXREF (Database cross-reference) module Project Plan as an example.

### 3. Write the script file that creates the data table.

You must modify the existing script file that creates the SeqHound data tables (this step is only for ODBC SeqHound). This file is called *seqhound.sql* and is located in *./slri/seqhound/sql/seqhound.sql*

#### Add the line

```
DROP TABLE SEQHOUND.DBXREF_tablename;
```

near the top of the file. Every time this script is run, it will destroy any pre-existing data tables that belong to the module.

Add the lines that describe the table(s) that are a part of the module so the script creates the new tables belonging to the module. See the example file "*seqhound.sql*".

Note that CodeBase tables do not require a script to be made; these tables are created by the function *InitCodeBase* in the *<module\_name>\_cb.c* file.

### 4. Write ASN.1 structure(s) that corresponds to data table descriptions.

The file *./slri/seqhound/asn/slristruc.asn* must be modified to contain a description of tables in the module. See the example structure *StDbXref* in the *slristruc.asn* file.

The shell script "*./slri/seqhound/asn/makeasn*" calls *asntool* which auto generates *slristruc.h*, *objsslristruc.h* and *objslristruc.c* files. These files contain functions that allow one to allocate and free memory for a structure corresponding to a database record. See the *makeasn* file.

### 5. Step 5: Design new functions for the Database code layer

The DB layer contains C functions that create and retrieve records from tables in the module. In the case of the DBXREF module, the tables are populated by a parser written in PERL so there are no functions listed that write to the tables. The requirement for a certain Perl modules is however noted in the project plan; these modules allow Perl to converse with a database.

The DB layer will have three files that were auto generated from the preceding step.

- a) *./slri/seqhound/include/slristruc.h*
- b) *./slri/seqhound/include/objsslristruc.h*
- c) *./slri/seqhound/src/objsslristruc.c*

These files will be placed in this location by the *makeasn* script.

In addition, DB layer will require that two new files be made.

- a) *./slri/seqhound/include/<module\_name>\_odbc.h*

This will contain function prototypes and comments for functions in

b) *./slri/seqhound/src/<module\_name>\_odbc.c*

This will contain at least one functions called “Search<module\_name>” that retrieves records from the database. Other functions that read or write to the database may be included in this file.

The example function “SearchDBXREF” (see *dbxref\_odbc.h* and *dbxref\_odbc.c*) takes as input a pointer to a structure called StDbXref. This structure has the same fields as the DBXREF table. This structure is described in ASN.1 and functions to allocate and free memory for this structure are auto generated using ASNtool. Any row in a DBXREF table that contains field values matching anyone of the field values passed to “SearchDBXREF” will be returned in a linked list of Valnodes that contain pointers to StDBXref structures.

6. Make changes to existing code to accommodate changes to the database code layer.

You must modify the existing function called `InitCodeBase` so that it handles tables that are new to the module. This function is found in two locations:

- a) *./slri/seqhound/src/intrez\_cb.c* is the file that supports a CodeBase database backend to SeqHound. `InitCodeBase()` needs to be changed to open the code base data files that belong to the new module. See the example *intrez\_cb.c* file.
- b) *./slri/seqhound/src/intrez\_odbc.c* is the file that supports a supports an ODBC database backend to SeqHound. `InitCodeBase()` need to contain the necessary code to establish a connection to the Database Server. See the example *intrez\_odbc.c* file.

The function `InitCodeBase()` under *intrez\_odbc.c* has a function call:

```
GetAppParam(intrez", "datab", "db2alias", NULL,
(Char*)dsn, sizeof(char) * 10) <= 0 )
```

that retrieves database connection information from the *./slri/seqhound/config/intrezrc* configuration file.

## Parser layer

7. Design the parser layer.

Parsers are generally written in C or Perl. A separate script is written to download some file from an external ftp site. The parser takes this file as input and uses it to populate a set of data tables belonging to the module.

Pseudocode for parsers should be documented in the project plan.

Parser layer code is located in

*./slri/seqhound/<module\_name>/[parser\_name]*

for example

*./slri/seqhound/dbxref/dbxref\_parser\_sp.pl*

Finally, by project end, all parsers must be documented according to the examples given in the SeqHound manual. See the example parser description for “mother”.

## Local API layer (Query layer)

### 8. Design the Local API layer

Design the local API local query layer. This layer will consist of three files

a) `./slri/seqhound/src/[Module name]_query.c`

This file contains all of the API functions that query the module's tables as well as auxiliary functions (if any). Note that this naming convention is not followed by API calls that belong to the core module of SeqHound; code for these local API calls that query core module tables is in `./slri/seqhound/src/intrez.c`.

b) `./slri/seqhound/include/[Module name]_query.h`

This file contains function prototypes and comments for auxiliary functions (if any) that may be used by the module's parsers, API functions or other applications specific to the module.

c) `./slri/seqhound/include/seqhound.h`

This is where all publicly available API functions are defined. Note that the local and the remote API's use the same header file. ALL API functions for ALL modules are defined in this header. This file already exists and must be simply modified.

Examples are:

**`./slri/seqhound/src/dbxref_query.c`**

**`./slri/seqhound/include/dbxref_query.h`**

**`./slri/seqhound/include/seqhound.h`**

An example of an API function is a function(s) to retrieve data base cross-references given a source record (`SHoundDBXREFGetDBXrefListBySourceRecord`). See the example in the `dbxref_query.c` file. This is a local API call. Notice the naming convention: 'Shound' followed by the module name 'GODB' followed by the actual API function name.

In the example function note the line

```
if (!SHoundModule("godb"))
```

This checks the SeqHound configuration file to make certain that the build of SeqHound actually includes this module. All API function calls must have an analogous check.

---

Note that this function calls a database layer function called `SearchDBXREF`.

---

## CGI layer

### 9. Design the CGI layer

The SeqHound cgi layer that supports the remote API is contained in only one file

**`./slri/seqhound/cgi/seqrem.c`**

There is no header file for `seqrem.c` since all functions are defined before "`Main()`" and `Main` only calls functions in this file.

This file must be edited to include cgi support for the remote API calls for the new module. See the examples in the *seqrem.c* file. See the example function *SeqHoundDBXREFGetDBXrefListBySourceRecord*.

Note the naming convention: SeqHound followed by the module name followed by the same function name used by the local API function.

## Remote API layer

### 10. Design the remote API layer

The remote API for SeqHound supports 4 languages. So there are four files that must be modified to include new API functions for the new module.

For C:	<i>./slri/seqhound/src/seqhoundapi.c</i>
For C++:	<i>./slri/seqhound/src_cxx/SeqHound.c</i>
For Java:	<i>./slri/seqhound/java/SeqHound.java</i>
For Perl:	<i>./slri/seqhound/perl/SeqHound.pm</i>

The function names will be exactly the same as those listed in the local API layer (for example; *SHoundDBXREFGetDBXrefListBySourceRecord*).

Examples for the C remote API are given in the *seqhoundapi.c* file.

### 11. Modify the seqhound config file.

New modules require that another entry be made in the *.intrezrc* file. This setting allows the SeqHound administrator to indicate whether any given SeqHound module has been built. The function *SHoundModule()* will look at these config file settings every time a local API call is made to determine if the module is present.

Modify the *./slri/seqhound/config/intrezrc* according to the example. Analogous additions must be made to the *./slri/seqhound/config/intrez.ini* file for Windows platforms.

Modify the *SHoundModule()* function in *./slri/seqhound/src/intrez\_cfg.c* file to support the new module. Follow the example in this file.

### 12. Modify/create make files to support the new module.

Modify the following files.

*./slri/seqhound/seqhound\_odbc.mk*

SEQH\_SRC\_ODBC  
SEQH\_OBJ\_OODBC

*./slri/seqhound/seqhound.mk*

SEQH\_SRC\_COM  
SEQH\_OBJ\_COM  
SEQH\_ODBC\_COM



***./slri/seqhound/seqhound\_rem.mk***

***./slri/seqhound/seqhoundrem.mk??***

***./slri/seqhound/src/make.shoundlocllib***

***./slri/seqhound/src/make.shoundremlib***

Create the following make files for any parsers written in C.

***./slri/seqhound/<module\_name>/make.[parser\_name]***

see the example in

***./slri/seqhound/locuslink/make.llparser***

### 13. Design regression tests

Regression tests are based on the CuTest C Unit Testing Framework.

(see <http://cutest.sourceforge.net/>).

One file must be modified to support tests for the new module. Follow the examples in:

***./slri/seqhound/test/regresion/main.c***

And a new file must be created that contains the actual test functions. See the examples in the test driver for the database layer for the DBXREF module:

***./slri/seqhound/test/regresion/dbxref\_odbc\_driver.c***

Examples of some test cases Function calls.

```
void testDBXREF_GetObjectIDbyAcc(CuTest *tc){...}
```

Another driver contains test functions for the

***./slri/seqhound/test/regresion/dbxref\_query\_driver.c***

Finally, the following file must be modified to accommodate the new module test drivers. Follow the examples in:

***./slri/seqhound/test/regresion/make.test\_driver***

### 14. Design test cases

These test cases refer to tests of the local and remote API functions relevant to the module.

### 15. Code test and debug

### 16. Finish documentation

### 17. Compile SeqHound code in test environment

### 18. Build SeqHound db in test environment

### 19. Check in code

### 20. Pass the project on to delivery team (Test Dev/Systems Dev/Software Training)

### 21. Build module in production (SEQHOUND ADMIN)

### 22. Update data tables (SEQHOUND ADMIN)

- 23. Update seqrem (SEQHOUND ADMIN)
- 24. Update docs (SEQHOUND ADMIN)
- 25. Update website (SEQHOUND ADMIN)

## 7. Appendices

*Example GenBank record in ASN.1 format*

*Example SwissProt record in ASN.1 format*

*Example EMBL record in ASN.1 format*

*Example PDB record in ASN.1 format*

*Example Biostruc in ASN.1 format*

*GO background material*

**Example GenBank record**

```

Seq-entry ::= set {
  class nuc-prot ,
  descr {
    title "Vairimorpha necatrix largest subunit of RNA polymerase II
(RPB1)
    gene, complete cds." ,
    source {
      org {
        taxname "Vairimorpha necatrix" ,
        db {
          {
            db "taxon" ,
            tag
              id 6039 } } , <=====TAXGI/taxid
        orgname {
          name
            binomial {
              genus "Vairimorpha" ,
              species "necatrix" } } ,
          lineage "Eukaryota; Fungi; Microsporidia; Burenellidae;
Vairimorpha" ,
          gcode 1 ,
          mgcode 1 ,
          div "INV" } } } ,
    create-date
      std {
        year 1998 ,
        month 12 ,
        day 10 } ,
    pub {
      pub {
        sub {
          authors {
            names
              std {
                {
                  name
                    name {
                      last "Hirt" ,
                      first "R" ,
                      initials "R.P." } } ,
                  {
                    name
                      name {
                        last "Healy" ,
                        first "B" ,
                        initials "B." } } } } ,
          affil
            std {
              affil "The Natural History Museum" ,
              div "Zoology" ,
              city "London" ,
              country "UK" ,
              street "Cromwell Road" ,
              postal-code "SW7 5BD" } } } ,
          medium email ,
          date
            std {
              year 1998 ,
              month 4 ,
              day 16 } } } } ,
    update-date
      std {
        year 1999 ,
        month 2 ,
        day 4 } ,
    pub {
      pub {
        article {

```

```

        title {
            name "Microsporidia are related to Fungi: evidence from the
largest subunit of RNA polymerase II and other proteins." } ,
        authors {
            names
            std {
                {
                    name
                    name {
                        last "Hirt" ,
                        initials "R.P." } } ,
                    {
                        name
                        name {
                            last "Logsdon" ,
                            initials "J.M." ,
                            suffix "Jr." } } ,
                    {
                        name
                        name {
                            last "Healy" ,
                            initials "B." } } ,
                    {
                        name
                        name {
                            last "Dorey" ,
                            initials "M.W." } } ,
                    {
                        name
                        name {
                            last "Doolittle" ,
                            initials "W.F." } } ,
                    {
                        name
                        name {
                            last "Embley" ,
                            initials "T.M." } } } ,
            affil
            str "Department of Zoology, The Natural History Museum,
London
SW7 5BD, United Kingdom." } ,
            from
            journal {
                title {
                    iso-jta "Proc. Natl. Acad. Sci. U.S.A." ,
                    ml-jta "Proc Natl Acad Sci U S A" ,
                    issn "0027-8424" ,
                    name "Proceedings of the National Academy of Sciences
of the
United States of America." } ,
                imp {
                    date
                    std {
                        year 1999 ,
                        month 1 ,
                        day 19 } ,
                    volume "96" ,
                    issue "2" ,
                    pages "580-585" ,
                    language "eng" } } ,
                ids {
                    pubmed 9892676 ,
                    medline 99110933 } } ,
                muid 99110933 ,
                pmid 9892676 } } } ,
            seq-set {
                <=====beginning of bioseq
                <=====look down for end of bioseq(see ASNDB/asn1)
                seq {
                    id {
                        genbank{<=====ACCDB/DB
                        name "AF060234" ,<===== =====ACCDB/name

```

```
accession "AF060234" ,<=====ACCDB/accession
version 1 } ,<=====ACCDB/version
gi 4001823 } , <=====Many tables/GI
descr {
  molinfo {
    biomol genomic } } ,
  inst {
    repr raw ,
    mol dna ,
    length 5019 ,
    seq-data
      ncbi2na
'171B6C0C434EFC0FBDC301F7E3FFFF3F396FF3FD7FFFF355C03BF8E03EC
10020DDB4350F6BF3FDD48203080352EDB5030D4578053A00EAF57037AEBF0C8FC00EA4
71202A7
FFCEF4D7B8008C3FF7E57AD3FE944F83C1C250EF53BEB4E18034200DFA0EEFEFFCFBDC
87C03CB
1000DF0808FE0FFB3A0CFDC21C02EFE60AE03EA803ABF129EE80C2497B0C0008283BF8
C9FF380
A208202E3AC0B0FF83A2022F4C33DE0823EC0E08E7BBFFABFE3482FC70978BA7CFF87BD
FF2F5D7
5F7B0A5CB3ED3A0A0EF889208E3F874C0FA48CCB009C31F37C203383C82A917AD3B2C2A
3CE043C
750FD33E510E3E10E130BA5051291F4802EA657F0024FDE761E029020A98B5AA03F0EA0
02AC8FD
2E52B7B0F1978D60CFDEC820B68BD7D223E608D447F5480F3075FF0CF8597C701FB740E
A5038B1
56A983CEC3080E1A1020C87C0FD0CAA233C2F2082BCECB622113908688ECBFCFC32105F
7F1100E
24E3A74FFB58B4E829007FC8703FB7EEDD95CC19A3FE1AA180E0DF44E5902F104B0A683
C820F3B
F2C2C042CF0B510DC3014B0EA8CB1084BF049DC21FFC4F188F4FFF8DB22010E43CF33DE
C0CF0C3
CE2FC78F6D80F0F3874E3874FEB0C3F131E0837D033CE00CF03F549CFDF35C203EE87AC
040DF0B
C4FF140C43F30E80370E244182087E803B608F7C6D30F203A6037C2E83CF842027B6AD0
4429AFC
3D130F90387FAD7848B87EFD7E38E748038E0DFCFD9C430197FDCFA0FAE190C92102010
E2D0B10
20B32012702042C0E034CB4091000301C808F14ACED0E22227F80B42C0F13FE0C050884
FDE99B4
9F4037F8BFFB0C138A18EB5E928D00A74FCC0CFD10B449FBF2B4100EC828008357F6BFC
3CC8DFC
513FCB0268F3DEA037228FEC8032F3B00A0F1D7820F7DFD39CEA2B220ADC3E33E73C012
4804A33
3D020872C0243A08E71ED1DF8C8DEC228923A3F3F3433831A208EBFE3947FFC80E40038
74E386E
9C702386EDFF003F93F2C84EFC48FE0FD90C000803BC48D03F103CF871E3B03F1200FF3
38E0FE2
EBC0E00EC000CE000E0CF9CB53B0FD40830C0FC98FCC0FE3EC802884FD56CFC3FC8C7F0
00FC320
0FC50F083F3C3E033C300CC3F3630228FF0383303CD3C827CC3E8FF0080F03F00FF00D0
C3F7503
80EB291FE9A7437B2A81494540E1FC0C7F53F24A2CD240CF18EA2C508F0080F303BE600
33C21D7
ECE08FCFC02353F0C01FC80E9C200D40B83C83FD233002DDEE0FCB80FCF3857B8C8211F
7300823
023FEF4A0CFF8FD5E3813F23FCB00E5C0FCD308F003E120F03EB0B003F80F200CFB002F
C4E297F
5C33FD4CC3C8D238032D20FC30C20C23B3F4DFE0C30CE833CC3F10CC00CCF03F023CEAF
30C0342
02FFCC2E08C08408E2EB3F12123A2FECC2203DDD3503B20A13FAC1F70E3F0380CB20BF
CA0C827
9C888437E0E0F87F2F321A03A0BCED0D329134971E9E3B8E10E0A8C5F12B304213A2E0C
0BA8FEA
E47019934FE0807BA33FC721927F2C920033B0402BCC488330E3A845C957FAA12832805
E7FE8EC
2C0F230273147F70148330F3820BE3194FCC44B52ED2005E2CC24BA01E895A7CFCB20A0
3633994
0042FCCF75862561F32D4189504CCB5462541332D4109507C495842541F32D4189585C7
4504D94
DB32D610B58BC4B5042562F12D4109589CCB5042D4DF3254109537C4B504D16DF12D413
```

```

45B7C4B
504256DF32D41095B7CCB504256DF32D41B18C38600801C3280600A024300CF0CB0000F
C00FFFF
FFEC03FFCCD035CCF8DCCBFC33C030304'H } ,
  annot {
    {
      data
      ftable {
        {
          data
          gene {
            locus "RPB1" } ,
          location
          int {
            from 100 ,
            to 4917 ,
            id
            gi 4001823 } } } } } ,

```

<=====the first bioseq ends here  
see ASNDB/asn1

<=====next bioseq begins here

```

seq {
  id {
    genbank {
      accession "AAD12604" ,
      version 1 } ,
      gi 4001824 } ,
  descr {
    molinfo {
      biomol peptide ,
      tech concept-trans-a } ,
      title "largest subunit of RNA polymerase II; RPO21 [Vairimorpha
necatrix]" } ,
  inst {
    repr raw ,
    mol aa ,
    length 1605 ,
    topology not-set ,
    seq-data
    ncbieaa
    "MFDEIVTKRISSIQFGLFSPEEIRKSSVVQIIHPETMENGFPKSGGLIDLKMGTTTERAF
LCSSCEKDNFSCPGHFGHIELTKPMFHVGYMTKIKKILECVCFCYCSRLKISTKNLKKDLNFWNISKTKSV
CEGEIGE
NGFTGCGNKQPVIKKEGMSLIAFMKGEEESDGKVIILNGERVHNILKKIVNEDAVFLGFDQKFTKPEWLILT
VLLVPPP
SVRPSIVMEGMLRAEDDLTHKLADIVKANTYLLKKYELEGAPGHVVRDYEQLLQFHIATMIDNDISGQPQAL
QKSGRPL
KSISARLKKGEGVRGNLMGKRVDFSARSVITPDPNISVEEVGVPSEIAKIHTFPEIITPFNIDRLTKLVS
NGPNEYF
GANYVIRNDGQRIDLNFNRGDIKLEEGYVVERHMQDGDVVLNFRQPSLHKMSMMAHFVRVMEGKTFRLNLS
CVSPYNA
DFDGDEMNLHMPQSYNSKAELEELCLVSKQVLSPOQSNKPMGIVQDSLALRLFTLRDSFFDRRETQMQLLY
SVNINNY
EFTDSSKLIMTHDDSFNNLHTEESSNIMKILNFPAISYPKPLWTGKQILSYILPNTIYNGKSNEHNEEDL
ENVEDSY
VIIRNGEILSGIIDKKAVGSTQGGLIHIIANDFGPDRVTCFFDDAQKMMNLYFATINAFSIGIGDAIADKE
TMSQVQR
SIETAKEQVNEIIVKAQKNKLERLPGMSMRESFESQVNYILNKPRDISGASASKSLFCNNMRTMVLGSK
GSFINIS
QVTACLQQNVGKRIPIFGFNYSRSLPHFSKADYSGKSRGFVENSYVKGITPEEFFFFHAMGGREGLIDIAIK
TAETGYI
QRRLVKAMEDATVTLDRSVRGADGFYQYEGEDGFDTFLEMQKMTHTDDVATKDDVSFKNLHLVDMFTDL
NFAIKKE
NVTQIYKLLTTDVNLQKILYDEFEFLNENVKKYKMNIASPCNFQRIINLAIYKFCRKGDISPYLILDT
LKNLIEN
LPIKNLLIEILIKYNLSIKRILNEYKLSLEAYNWILKEIKFKILKSIISPNEVMGTLAAQSVGEPATQMTL
NTFHLAG
VSANITMGVPRLEIINVAKNIKTPCMKIYKDPFNKTLEMAKKIQSELEFSDIKSLCEFSEIYYDPVIED
TSIKEDK

```

Version 3.2



**Example SwissProt record**

```

Seq-entry ::= seq {
  id {
    swissprot {
      name "RPB1_YEAST" , <=====ACCDDB/name
      accession "P04050" } , <=====ACCDDB/accession
      gi 2507347 } ,
    descr {
      title "DNA-DIRECTED RNA POLYMERASE II LARGEST SUBUNIT (B220)." ,
      comment "-----"
    }
  }
  -----
  ---This SWISS-PROT entry is copyright. It is produced through
  a~collaboration
  between the Swiss Institute of Bioinformatics and~the EMBL outstation
  - the
  European Bioinformatics Institute.~The original entry is available
  from
  http://www.expasy.ch/sprot~and
  http://www.ebi.ac.uk/sprot~-----
  -----
  comment "[FUNCTION] DNA-DEPENDENT RNA POLYMERASE CATALYZES THE
  TRANSCRIPTION OF DNA INTO RNA USING THE FOUR RIBONUCLEOSIDE
  TRIPHOSPHATES AS
  SUBSTRATES." ,
  comment "[CATALYTIC ACTIVITY] N NUCLEOSIDE TRIPHOSPHATE = N
  PYROPHOSPHATE
  + RNA(N)." ,
  comment "[SUBUNIT] RNA POLYMERASE II CONSISTS OF 12 DIFFERENT
  SUBUNITS.
  THIS SUBUNIT IS THE LARGEST COMPONENT OF RNA POLYMERASE II." ,
  comment "[SUBCELLULAR LOCATION] NUCLEAR." ,
  comment "[PTM] THE TANDEM 7 RESIDUES REPEATS CAN BE HIGHLY
  PHOSPHORYLATED.
  THE PHOSPHORYLATION ACTIVATES POL2." ,
  comment "[MISCELLANEOUS] THREE DISTINCT ZINC-CONTAINING RNA
  POLYMERASES
  ARE FOUND IN EUKARYOTIC NUCLEI: POLYMERASE I FOR THE RIBOSOMAL RNA
  PRECURSOR,
  POLYMERASE II FOR THE MRNA PRECURSOR, AND POLYMERASE III FOR 5S AND
  TRNA
  GENES." ,
  comment "[SIMILARITY] BELONGS TO THE RNA POLYMERASE BETA' CHAIN
  FAMILY." ,
  sp {
    class standard ,
    extra-acc {
      "Q12364" ,
      "Q92315" } ,
    seqref {
      gi 4397 ,
      gi 4398 ,
      gi 1419218 ,
      gi 1419221 ,
      gi 1431216 ,
      gi 1431217 ,
      gi 886080 ,
      gi 886082 ,
      gi 2144431 } ,
    dbref {
      {
        db "SGD" ,
        tag
        str "L0001744" } ,
      {
        db "PFAM" ,
        tag
        str "PF00623" } ,
    }
  }
}

```

```

    {
      db "PROSITE" ,
      tag
      str "PS00115" } } ,
keywords {
  "Transferase" ,
  "DNA-directed RNA polymerase" ,
  "Transcription" ,
  "Zinc" ,
  "Repeat" ,
  "DNA-binding" ,
  "Nuclear protein" ,
  "Phosphorylation" ,
  "Zinc-finger" } ,
created
std {
  year 1986 ,
  month 11 ,
  day 1 } ,
sequpd
std {
  year 1997 ,
  month 11 ,
  day 1 } ,
annotupd
std {
  year 1999 ,
  month 7 ,
  day 15 } } ,
create-date
std {
  year 1986 ,
  month 11 ,
  day 1 } ,
update-date
std {
  year 1999 ,
  month 7 ,
  day 15 } ,
source {
  org {
    taxname "Saccharomyces cerevisiae" ,
    common "baker's yeast" ,
    db {
      {
        db "taxon" ,
        tag
        id 4932 } } ,
    orgname {
      name
      binomial {
        genus "Saccharomyces" ,
        species "cerevisiae" } ,
      lineage "Eukaryota; Fungi; Ascomycota; Saccharomycotina;
Saccharomycetes; Saccharomycetales; Saccharomycetaceae; Saccharomyces"
    ,
      gcode 1 ,
      mgcode 3 ,
      div "PLN" } } } ,
molinfo {
  biomol peptide ,
  completeness complete } ,
pub {
  pub {
    gen {
      serial-number 1 } ,
    muid 85282617 ,
    article {
      title {
        name "Extensive homology among the largest subunits of
eukaryotic
and prokaryotic RNA polymerases." } ,

```

```

authors {
  names
  std {
    {
      name
      name {
        last "Allison" ,
        initials "L.A." } } ,
    {
      name
      name {
        last "Moyle" ,
        initials "M." } } ,
    {
      name
      name {
        last "Shales" ,
        initials "M." } } ,
    {
      name
      name {
        last "Ingles" ,
        initials "C.J." } } } } ,
from
  journal {
    title {
      iso-jta "Cell" ,
      ml-jta "Cell" ,
      issn "0092-8674" ,
      name "Cell." } ,
    imp {
      date
      std {
        year 1985 ,
        month 9 } ,
      volume "42" ,
      issue "2" ,
      pages "599-610" ,
      language "eng" } } ,
    ids {
      pubmed 3896517 ,
      medline 85282617 } } ,
  pmid 3896517 } ,
  comment "SEQUENCE FROM N.A.~STRAIN=A364A" } ,
pub {
  pub {
    gen {
      serial-number 2 } ,
    muid 97127826 ,
    article {
      title {
        name "Analysis of a 26,756 bp segment from the left arm of
yeast
chromosome IV." } ,
      authors {
        names
        std {
          {
            name
            name {
              last "Wofl" ,
              initials "S." } } ,
          {
            name
            name {
              last "Hanemann" ,
              initials "V." } } ,
          {
            name
            name {
              last "Saluz" ,
              initials "H.P." } } } } ,

```

```

        affil
        str "Hans-Knoll-Institut fur Naturstoff-Forschung,
Department of
Cell and Molecular Biology, Jena, Germany." } ,
    from
    journal {
        title {
            iso-jta "Yeast" ,
            ml-jta "Yeast" ,
            issn "0749-503X" ,
            name "Yeast (Chichester, England)" } ,
        imp {
            date
            std {
                year 1996 ,
                month 12 } ,
            volume "12" ,
            issue "15" ,
            pages "1549-1554" ,
            language "eng" } } ,
        ids {
            pubmed 8972577 ,
            medline 97127826 } } ,
        pmid 8972577 } ,
        comment "SEQUENCE FROM N.A.~STRAIN=S288C / FY1679" } ,
    pub {
        pub {
            gen {
                serial-number 3 } ,
            muid 95377607 ,
            article {
                title {
                    name "The gene encoding the biotin-apoprotein ligase of
Saccharomyces cerevisiae." } ,
                authors {
                    names
                    std {
                        {
                            name
                            name {
                                last "Cronan" ,
                                initials "J.E." ,
                                suffix "Jr." } } ,
                        {
                            name
                            name {
                                last "Wallace" ,
                                initials "J.C." } } } ,
                    affil
                    str "Department of Microbiology, University of Illinois,
Urbana
6180, USA." } ,
                from
                journal {
                    title {
                        iso-jta "FEMS Microbiol. Lett." ,
                        ml-jta "FEMS Microbiol Lett" ,
                        issn "0378-1097" ,
                        name "FEMS microbiology letters." } ,
                    imp {
                        date
                        std {
                            year 1995 ,
                            month 8 ,
                            day 1 } ,
                        volume "130" ,
                        issue "2-3" ,
                        pages "221-229" ,
                        language "eng" } } ,
                    ids {
                        pubmed 7649444 ,
                        medline 95377607 } } ,

```

Version 3.2

```

        to 82 ,
        id
        gi 2507347 } ,
    exp-ev not-experimental } ,
{
    data
        region "Domain" ,
        comment "CARBOXYL-TERMINAL 7-RESIDUE REPEATS." ,
        location
            int {
                from 1543 ,
                to 1718 ,
                id
                gi 2507347 } ,
        exp-ev experimental } ,
{
    data
        region "Variant" ,
        comment "MISSING (IN STRAIN A364A)." ,
        location
            int {
                from 1652 ,
                to 1658 ,
                id
                gi 2507347 } ,
        exp-ev experimental } ,
{
    data
        region "Conflict" ,
        comment "A -> V (IN REF. 1)." ,
        location
            pnt {
                point 1513 ,
                id
                gi 2507347 } ,
        exp-ev experimental } ,
{
    data
        region "Conflict" ,
        comment "G -> A (IN REF. 1)." ,
        location
            pnt {
                point 1523 ,
                id
                gi 2507347 } ,
        exp-ev experimental } ,
{
    data
        region "Conflict" ,
        comment "T -> M (IN REF. 1)." ,
        location
            pnt {
                point 1600 ,
                id
                gi 2507347 } ,
        exp-ev experimental } ,
{
    data
        gene {
            locus "RPB1" ,
            syn {
                "RPO21" ,
                "RPB220" ,
                "SUA8" ,
                "YDL140C" ,
                "D2150" } } ,
        location
            int {
                from 0 ,
                to 1732 ,
                id
                gi 2507347 } } ,

```

```
{
  data
  prot {
    name {
      <=====ACCDB/title
      "DNA-DIRECTED RNA POLYMERASE II LARGEST SUBUNIT" } ,
    ec {
      "2.7.7.6" } } ,
  location
  int {
    from 0 ,
    to 1732 ,
    id
    gi 2507347 } } } } }
```

**Example EMBL record**

```

Seq-entry ::= set {
  level 1 ,
  class nuc-prot ,
  descr {
    pub {
      pub {
        gen {
          cit "Unpublished" ,
          authors {
            names
            std {
              {
                name
                name {
                  last "Drebot" ,
                  initials "M.A." } } ,
              {
                name
                name {
                  last "Jansma" ,
                  initials "D." } } ,
              {
                name
                name {
                  last "Himmelfarb" ,
                  initials "H.J." } } ,
              {
                name
                name {
                  last "Friesen" ,
                  initials "J.D." } } } } ,
          title "Suppressors of yeast RNA polymerase II mutations
belong to a
family of gene products that interact with a protein kinase" } } } ,
    pub {
      pub {
        sub {
          authors {
            names
            std {
              {
                name
                name {
                  last "Jansma" ,
                  initials "D." } } } ,
          affil
          str "David Jansma, Genetics, Hospital for Sick Children,
555
University, Avenue, Toronto, Ontario, M5G 1X8, Canada" } ,
          medium other ,
          date
          std {
            year 1992 ,
            month 7 ,
            day 28 } } } } ,
          create-date
          std {
            year 1992 ,
            month 12 ,
            day 11 } ,
          update-date
          std {
            year 1993 ,
            month 3 ,
            day 12 } ,
          source {
            org {
              taxname "Saccharomyces cerevisiae" ,

```



```

common "baker's yeast" ,
db {
  {
    db "taxon" ,
    tag
    id 4932 } } ,
orgname {
  name
  binomial {
    genus "Saccharomyces" ,
    species "cerevisiae" } ,
  mod {
    {
      subtype isolate ,
      subname "Mating type a" } } ,
  lineage "Eukaryota; Fungi; Ascomycota; Saccharomycotina;
Saccharomycetes; Saccharomycetales; Saccharomycetaceae; Saccharomyces"
,
  gcode 1 ,
  mgcode 3 ,
  div "PLN" } } ,
subtype {
  {
    subtype chromosome ,
    name "7" } } } } ,
seq-set {
  seq {
    id {
      embl {
        name "SCSPM2" ,<=====ACCDDB/name
        accession "Z14128" ,<=====ACCDDB/accession
        version 1 } ,<=====ACCDDB/version
        gi 287914 } ,
      descr {
        title "S.cerevisiae spm2+ gene for spm2+ protein" ,
        embl {
          div fun ,
          creation-date
          std {
            year 1992 ,
            month 12 ,
            day 11 } ,
          update-date
          std {
            year 1993 ,
            month 3 ,
            day 12 } ,
          keywords {
            "CDC68/SPT16 protein" ,
            "spm2+ gene" ,
            "spm2+ protein" } ,
          xref {
            {
              dbname
              name "SGD" ,
              id {
                str "L0001891" ,
                str "SIP2" } } ,
              {
                dbname
                code swissprot ,
                id {
                  str "P34164" ,
                  str "SIP2_YEAST" } } } } ,
            molinfo {
              biomol genomic } } ,
            inst {
              repr raw ,
              mol dna ,
              length 2032 ,
              seq-data
              ncbi2na

```

```

'00F965EC6C171E0103E3BFD06B54022FC17F9A55FFCDD14FFD770607B4E
0107C933066C7C4C8C8F0323053933470724DB4E0C91BF973183E69693ADFF33E757E85
8B023F1
7E6FF68DB4037FCA80A303114423D7A80E00B3780C3EE7FE82AC78892E0BF73AB1C610B
4D49500
024041000BB6E45CD38B86F2200152439D0A798164203A39EE24202F1E0789F0303BD23
D50866A
925F42A2A373C50000361BCF9E6E38A18951CE502737BCE8079EE847877AB7DB71C98E3
822A233
CF9521848950908E52D78E38D9D294F7D952228A841240C6E50209F6A2B50B80D089D3C
3AF56E8
8C2BA490AEBD00BF1B84A74F4503A28038DAFE3178F78430E9D3F4EE09C29E7D4A444C8
F48FCCB
A30E27C8BCB8FD7951241E348EA01F6D0F1320B4A4148000541C180230AD428248F5398
5D51498
D8D743E77903EB0A35218FFAE1A3310AFD38237B45485D6FA0CC41213D596EFC7856DED
3A0631C
F317E8DB4902431E31B4E9E114550F175D2FA20EF37E0C0C739464214BD062041D299DE
54F5814
EEB9F051FAF1CB24F0910C47FBB25D4DBD8CC0480CED1523DFCC654C8B5DB0D6A3D71FF
C0A6CFB
333FFA49CE911C3F73F50FCBB0922332093304C38FC70EA74B1DD0E7CFBCD1014631C31
5A0B66E
37FFFFFF6803883651FF706020CF80DF0CFAE208A0C02C9C222D1A8700AC00BD273D41FC
75FDE33
E45FDB81E9F6ECFC2E0CE'H ,
    hist {
      replaces {
        date
        std {
          year 1993 ,
          month 6 ,
          day 11 } ,
        ids {
          gi 4525 } } } ,
    annot {
      {
        data
        ftable {
          {
            data
            pub {
              pub {
                muid 92017853 ,
                article {
                  title {
                    name "CDC68, a yeast gene that affects
regulation of
cell proliferation and transcription, encodes a protein with a highly
acidic
carboxyl terminus." } ,
                    authors {
                      names
                      std {
                        {
                          name
                          name {
                            last "Rowley" ,
                            initials "A." } } ,
                        {
                          name
                          name {
                            last "Singer" ,
                            initials "R.A." } } ,
                        {
                          name
                          name {
                            last "Johnston" ,
                            initials "G.C." } } } ,
                      affil
                      str "Department of Microbiology, Dalhousie
University, Halifax, Nova Scotia, Canada." } ,

```

```

from
  journal {
    title {
      iso-jta "Mol. Cell. Biol." ,
      ml-jta "Mol Cell Biol" ,
      issn "0270-7306" ,
      name "Molecular and cellular biology." }
  }
,
  imp {
    date
      std {
        year 1991 ,
        month 11 } ,
    volume "11" ,
    issue "11" ,
    pages "5718-5726" ,
    language "eng" } } ,
  ids {
    pubmed 1833637 ,
    medline 92017853 } } ,
  pmid 1833637 } } ,
  location
    int {
      from 1282 ,
      to 2031 ,
      id
        gi 287914 } } ,
  {
    data
      gene {
        locus "spm2+" } ,
    location
      int {
        from 398 ,
        to 1645 ,
        id
          gi 287914 } } } } } ,
seq {
  id {
    embl {
      accession "CAA78503" ,
      version 1 } ,
    gi 287915 } ,
  descr {
    title "spm2+ [Saccharomyces cerevisiae]" ,
    molinfo {
      biomol peptide } } ,
  inst {
    repr raw ,
    mol aa ,
    length 415 ,
    seq-data
      ncbieaa
      "MGTTTSHPAQKKQTTKKCRAPIMSDVREKPSNAQGCEPQEMDAVSKKVTELSLNKCSDS
      QDAGQPSREGSITKKKSTLLLRDEDEPTMPKLSVMETAVDTDGSSSTSDDEEGDIIAQTTPEPKQDASPDD
      DRSGHSS
      PREEGQQQIRAKEASGGPSEIKSSLMVPVEIRWQQGSKVYVTGSFTKWRKMIGLIPSDNNGSFHVKLRL
      LPGAHRF
      RFIVDNELRVSDFLPTATDQMGNFVNYIEVRQPEKNPTNEKIRSKEADSMRPPTSDRSSIALQIGKDPDDF
      GDGYTRF
      HEDLSRPPLLEYTTDIPAVFTDPSVMERYYYTLDRQQSNTDTSWLTPPQLPPQLENVILNKYYATQDQFNE
      NNSGALP
      IPNHVVLNHLVTSSIKHNTLCVASIVRYKQKYVTQILYTPPIESS" ,
    hist {
      replaces {
        date
          std {
            year 1993 ,
            month 6 ,
            day 11 } ,
        ids {
          gi 4526 } } } } ,

```

```

    annot {
    {
        data
        ftable {
        {
            data
            prot {
            name {
            "spm2+" } ,
            activity {
            "Wild-type version of SPM2,a dominant extragenic
            suppressor of some temperature-sensitive mutations in RPO21 and PRP4."
            } } ,
            location
            whole
            gi 287915 } } } } } ,
    seq {
    id {
    embl {
    accession "CAA78504" ,
    version 1 } ,
    gi 4388554 } ,
    descr {
    title "CDC68 /SPT16 [Saccharomyces cerevisiae]" ,
    molinfo {
    biomol peptide ,
    completeness partial } } ,
    inst {
    repr raw ,
    mol aa ,
    length 1 ,
    seq-data
    ncbieaa "M" } ,
    annot {
    {
        data
        ftable {
        {
            data
            prot {
            name {
            "CDC68 /SPT16" } } ,
            partial TRUE ,
            location
            whole
            gi 4388554 } } } } } } ,
    annot {
    {
        data
        ftable {
        {
            data
            cdregion {
            frame one ,
            code {
            id 1 } } ,
            product
            whole
            gi 287915 ,
            location
            int {
            from 398 ,
            to 1645 ,
            id
            gi 287914 } ,
            dbxref {
            {
                db "SWISS-PROT" ,
                tag
                str "P34164" } } } ,
            {
            data

```

```
    cdregion {
      frame one ,
      code {
        id 1 } } ,
    partial TRUE ,
    product
      whole
        gi 4388554 ,
    location
      int {
        from 2029 ,
        to 2031 ,
        id
          gi 287914 ,
        fuzz-to
          lim gt } ,
    cit
      pub {
        muid 92017853 } } } } }
```

**Example PDB record**

```

Seq-entry ::= set {
  class pdb-entry ,
  descr {
    pdb {
      deposition
      std {
        year 1992 ,
        month 4 ,
        day 3 } ,
      class "Isomerase(Intramolecular Oxidoreductse)" ,
      compound {
        "D-Xylose Isomerase (E.C.5.3.1.5) Mutant With Glu 186 Replaced
By Gln
(E186Q) Complex With Xylose And Mn" } ,
      source {
        "(Actinoplanes missouriensis) E186Q Mutant Gene Expressed In
(Escherichia coli)" } ,
      exp-method "X-Ray Diffraction" } ,
      comment "Revision History:~JUL 15 93 Initial Entry" ,
      pub {
        pub {
          sub {
            authors {
              names
              std {
                {
                  name
                  name {
                    last "Janin" ,
                    full "J.Janin" ,
                    initials "J." } } } } ,
            date
            std {
              year 1992 ,
              month 4 ,
              day 3 } } } } ,
        pub {
          pub {
            muid 92304915 ,
            article {
              title {
                name "Protein engineering of xylose (glucose) isomerase
from
Actinoplanes missouriensis. 1. Crystallography and site-directed
mutagenesis
of metal binding sites." } ,
              authors {
                names
                str {
                  "J.Jenkins" ,
                  "J.Janin" ,
                  "F.Rey" ,
                  "M.Chiadmi" ,
                  "H.van Tilbeurgh" ,
                  "I.Lasters" ,
                  "M.De Maeyer" ,
                  "D.Van Belle" ,
                  "S.J.Wodak" ,
                  "M.Lauwereys" ,
                  "P.Stanssens" ,
                  "N.T.Mrabet" ,
                  "J.Snauwaert" ,
                  "G.Matthyssens" ,
                  "A.-M.Lambeir" } } } ,
              from
              journal {
                title {

```

```

        iso-jta "Biochemistry" ,
        ml-jta "Biochemistry" ,
        issn "0006-2960" ,
        name "Biochemistry." } ,
    imp {
        date
        std {
            year 1992 ,
            month 6 ,
            day 23 } ,
        volume "31" ,
        issue "24" ,
        pages "5449-5458" ,
        language "eng" } } ,
    ids {
        pubmed 1610791 ,
        medline 92304915 } } ,
    pmid 1610791 } } ,
    pub {
        pub {
            muid 92304916 ,
            article {
                title {
                    name "Protein engineering of xylose (glucose) isomerase
from
Actinoplanes missouriensis. 2. Site-directed mutagenesis of the xylose
binding site." } ,
                authors {
                    names
                    str {
                        "A.-M.Lambeir" ,
                        "M.Lauwereys" ,
                        "P.Stanssens" ,
                        "N.T.Mrabet" ,
                        "J.Snauwaert" ,
                        "H.van Tilbeurgh" ,
                        "G.Matthyssens" ,
                        "I.Lasters" ,
                        "M.De Maeyer" ,
                        "S.J.Wodak" ,
                        "J.Jenkins" ,
                        "M.Chiadmi" ,
                        "J.Janin" } } ,
                    from
                    journal {
                        title {
                            iso-jta "Biochemistry" ,
                            ml-jta "Biochemistry" ,
                            issn "0006-2960" ,
                            name "Biochemistry." } ,
                        imp {
                            date
                            std {
                                year 1992 ,
                                month 6 ,
                                day 23 } ,
                                volume "31" ,
                                issue "24" ,
                                pages "5459-5466" ,
                                language "eng" } } ,
                            ids {
                                pubmed 1610792 ,
                                medline 92304916 } } ,
                            pmid 1610792 } } ,
                        pub {
                            pub {
                                muid 92304917 ,
                                article {
                                    title {
                                        name "Protein engineering of xylose (glucose) isomerase
from
Actinoplanes missouriensis. 3. Changing metal specificity and the pH

```

```

profile
  by site-directed mutagenesis." } ,
  authors {
    names
    std {
      {
        name
        name {
          last "van Tilbeurgh" ,
          initials "H." } } ,
      {
        name
        name {
          last "Jenkins" ,
          initials "J." } } ,
      {
        name
        name {
          last "Chiadmi" ,
          initials "M." } } ,
      {
        name
        name {
          last "Janin" ,
          initials "J." } } ,
      {
        name
        name {
          last "Wodak" ,
          initials "S.J." } } ,
      {
        name
        name {
          last "Mrabet" ,
          initials "N.T." } } ,
      {
        name
        name {
          last "Lambeir" ,
          initials "A.M." } } } ,
    affil
    str "Plant Genetic Systems N.V., Gent, Belgium." } ,
  from
  journal {
    title {
      iso-jta "Biochemistry" ,
      ml-jta "Biochemistry" ,
      issn "0006-2960" ,
      name "Biochemistry." } ,
    imp {
      date
      std {
        year 1992 ,
        month 6 ,
        day 23 } ,
      volume "31" ,
      issue "24" ,
      pages "5467-5471" ,
      language "eng" } } ,
    ids {
      pubmed 1610793 ,
      medline 92304917 } } ,
  pmid 1610793 } } ,
  pub {
    pub {
      muid 92172844 ,
      article {
        title {
          name "Arginine residues as stabilizing elements in
proteins." } ,
        authors {
          names

```



```

str {
  "N.T.Mrabet" ,
  "A.Van Den Broek" ,
  "I.Van Den Brande" ,
  "P.Stanssens" ,
  "Y.Laroche" ,
  "A.-M.Lambeir" ,
  "G.Matthijssens" ,
  "J.Jenkins" ,
  "M.Chiadmi" ,
  "H.van Tilbeurgh" ,
  "F.Rey" ,
  "J.Janin" ,
  "W.J.Quax" ,
  "I.Lasters" ,
  "M.De Maeyer" ,
  "S.J.Wodak" } } ,
from
journal {
  title {
    iso-jta "Biochemistry" ,
    ml-jta "Biochemistry" ,
    issn "0006-2960" ,
    name "Biochemistry." } ,
  imp {
    date
    std {
      year 1992 ,
      month 3 ,
      day 3 } ,
    volume "31" ,
    issue "8" ,
    pages "2239-2253" ,
    language "eng" } } ,
  ids {
    pubmed 1540579 ,
    medline 92172844 } } ,
  pmid 1540579 } } ,
pub {
  pub {
    muid 89184498 ,
    article {
      title {
        name "Structural analysis of the 2.8 A model of Xylose
isomerase
from Actinoplanes missouriensis." } ,
      authors {
        names
        std {
          {
            name
            name {
              last "Rey" ,
              initials "F." } } ,
          {
            name
            name {
              last "Jenkins" ,
              initials "J." } } ,
          {
            name
            name {
              last "Janin" ,
              initials "J." } } ,
          {
            name
            name {
              last "Lasters" ,
              initials "I." } } ,
          {
            name
            name {

```

```

        last "Alard" ,
        initials "P." } } ,
    {
        name
        name {
            last "Claessens" ,
            initials "M." } } ,
    {
        name
        name {
            last "Matthyssens" ,
            initials "G." } } ,
    {
        name
        name {
            last "Wodak" ,
            initials "S." } } } ,
    affil
    str "Laboratoire de Biologie Physicochimique, Universite
Paris
Sud, Orsay, France." } ,
    from
    journal {
        title {
            iso-jta "Proteins" ,
            ml-jta "Proteins" ,
            issn "0887-3585" ,
            name "Proteins." } ,
        imp {
            date
            std {
                year 1988 } ,
            volume "4" ,
            issue "3" ,
            pages "165-172" ,
            language "eng" } } ,
        ids {
            pubmed 3237716 ,
            medline 89184498 } } ,
        pmid 3237716 } } } ,
    seq-set {
        seq {
            id {
                pdb {
                    mol "9XIM" ,<=====ACCDB/name
                    chain 65 ,<=====ACCDB/chain
                    rel
                    std {
                        year 1992 , ,<=====ACCDB/release
                        month 4 ,
                        day 3 } } ,
                    gi 443580 } ,
                descr {
// record truncated

```

## Example Biostruc

An example of the ASN biostruc. Some data has been removed for the sake of brevity.

```

Biostruc ::= {
  id {
    mmdb-id 2 } ,
  descr {
    name "101D" ,
    pdb-comment "remark 3: Refinement." ,
    pdb-comment "remark 3: Program Nuclsq" ,
    pdb-comment "remark 3: Authors Westhof,Dumas,Moras" ,
    pdb-comment "remark 3: R Value 0.163" ,
    pdb-comment "remark 3: Free R Value 0.252" ,
    pdb-comment "remark 3: Number Of Reflections 2430" ,
    pdb-comment "remark 3: Resolution Range 8.0 - 2.25 Angstroms" ,
    pdb-comment "remark 3: Data Cutoff 2.0 Sigma(F)" ,
    pdb-comment "remark 3: Number Of Protein Atoms 0" ,
    pdb-comment "remark 3: Number Of Nucleic Acid Atoms 488" ,
    pdb-comment "remark 3: Number Of Solvent Atoms 33" ,
    pdb-comment "remark 3: Rms Deviations From Ideal Values (The Values Of" ,
    pdb-comment "remark 3: Sigma, In Parentheses, Are The Input Estimated" ,
    pdb-comment "remark 3: Standard Deviations That Determine The Relative" ,
    pdb-comment "remark 3: Weights Of The Corresponding Restraints)" ,
    pdb-comment "remark 3: Distance Restraints (Angstroms)" ,
    pdb-comment "remark 3: Sugar-Base Bond Distance 0.024(0.030)" ,
    pdb-comment "remark 3: Sugar-Base Bond Angle Distance 0.040(0.040)" ,
    pdb-comment "remark 3: Phosphate Bond Distance 0.026(0.040)" ,
    pdb-comment "remark 3: Phosphate Bond Angle Distance, H-Bond 0.057(0.050)" ,
    pdb-comment "remark 3: Plane Restraint (Angstroms) 0.014(0.020)" ,
    pdb-comment "remark 3: Chiral-Center Restraint (Angstroms3) 0.161(0.150)" ,
    pdb-comment "remark 3: Non-Bonded Contact Restraints (Angstroms)" ,
    pdb-comment "remark 3: Single Torsion Contact 0.093(0.100)" ,
    pdb-comment "remark 3: Multiple Torsion Contact 0.097(0.100)" ,
    pdb-comment "remark 3: Isotropic Thermal Factor Restraints (Angstroms2)" ,
    pdb-comment "remark 3: Sugar-Base Bond 4.282(6.000)" ,
    pdb-comment "remark 3: Sugar-Base Angle 4.990(6.000)" ,
    pdb-comment "remark 3: Phosphate Bond 5.693(6.000)" ,
    pdb-comment "remark 3: Phosphate Bond Angle, H-Bond 5.227(6.000)" ,
    pdb-comment "remark 101: Residue +c A 9 Has Br Bonded To C5." ,
    pdb-comment "remark 101: Residue +c B 21 Has Br Bonded To C5." ,
    pdb-comment "remark 105: The Protein Data Bank Has Adopted The Saccharide Chemists" ,
    pdb-comment "remark 105: Nomenclature For Atoms Of The DeoxyriboseRIBOSE MOIETY" ,
    pdb-comment "remark 105: Rather Than That Of The Nucleoside Chemists. The Ring" ,
    pdb-comment "remark 105: Oxygen Atom Is Labelled O4 Instead Of O1." ,
    pdb-comment "remark 106: The Hydrogen Bonds Between Base Pairs In This Entry Follow" ,
    pdb-comment "remark 106: The Conventional Watson-Crick Hydrogen Bonding Pattern." ,
    pdb-comment "remark 106: They Have Not Been Presented On Conect Records In

```

```

This" ,
  pdb-comment "remark 106: Entry." ,
  history {
    data-source {
      name-of-database "Protein Data Bank" ,
      version-of-database
      release-date
      std {
        year 1995 ,
        month 2 ,
        day 28 } ,
      database-entry-id
      other-database {
        db "PDB" ,
        tag
        str "101D" } ,
      database-entry-date
      std {
        year 1994 ,
        month 12 ,
        day 14 } } } ,
  attribution
  sub {
    authors {
      names
      std {
        {
          name
          name {
            last "Goodsell" ,
            full "D.S.Goodsell" ,
            initials "D.S." } } ,
        {
          name
          name {
            last "Kopka" ,
            full "M.L.Kopka" ,
            initials "M.L." } } ,
        {
          name
          name {
            last "Dickerson" ,
            full "R.E.Dickerson" ,
            initials "R.E." } } } } ,
    imp {
      date
      std {
        year 1994 ,
        month 12 ,
        day 14 } } } ,
  attribution
  gen {
    cit "To Be Published" ,
    authors {
      names
      std {
        {
          name
          name {
            last "Goodsell" ,
            full "D.S.Goodsell" ,
            initials "D.S." } } ,
        {
          name
          name {
            last "Kopka" ,
            full "M.L.Kopka" ,
            initials "M.L." } } ,
        {
          name
          name {
            last "Dickerson" ,

```

```

        full "R.E.Dickerson" ,
        initials "R.E." } } } } ,
    title "Refinement Of Netropsin Bound To Dna: Bias And Feedback In
Electron Density Map Interpretation" } ,
    attribution
    equiv {
        muid 85264810 ,
        article {
            title {
                name "Binding of an antitumor drug to DNA, Netropsin and
C-G-C-G-A-A-T-T-BrC-G-C-G." } ,
            authors {
                names
                std {
                    {
                        name
                        name {
                            last "Kopka" ,
                            initials "M.L." } } ,
                    {
                        name
                        name {
                            last "Yoon" ,
                            initials "C." } } ,
                    {
                        name
                        name {
                            last "Goodsell" ,
                            initials "D." } } ,
                    {
                        name
                        name {
                            last "Pjura" ,
                            initials "P." } } ,
                    {
                        name
                        name {
                            last "Dickerson" ,
                            initials "R.E." } } } } } ,
        from
        journal {
            title {
                iso-jta "J. Mol. Biol." ,
                ml-jta "J Mol Biol" ,
                issn "0022-2836" ,
                jta "J6V" } ,
            imp {
                date
                std {
                    year 1985 ,
                    month 6 ,
                    day 25 } ,
                volume "183" ,
                issue "4" ,
                pages "553-563" } } } } } ,
    chemical-graph {
        descr {
            name "Dna (5'-D(CpGpCpGpApApTpTp(Br)cpGpCpG)-3') Complexed With
Netropsin, Re-Refinement" ,
            pdb-class "Deoxyribonucleic Acid" ,
            pdb-source "Synthetic" ,
            assembly-type other } ,
        molecule-graphs {
            {
                id 1 ,
                descr {
                    name "A" ,
                    pdb-comment "SEQRES" ,
                    molecule-type dna ,
                    organism {
                        org {
                            taxname "synthetic construct" ,

```

```

db {
  {
    db "taxon" ,
    tag
    id 32630 } } ,
  orgname {
    name
    partial {
      {
        fixed-level other ,
        level "species" ,
        name "synthetic construct" } } ,
        lineage "artificial sequence" ,
        gcode 11 ,
        div "SYN" } } } } ,
seq-id
  gi 996094 ,
  residue-sequence {
    {
      id 1 ,
      name " 1 " ,
      residue-graph
      standard {
        biostruc-residue-graph-set-id
        other-database {
          db "Standard residue dictionary" ,
          tag
          id 1 } ,
        residue-graph-id 66 } } ,
    {
      id 5 ,
      name " 5 " ,
      residue-graph
      standard {
        biostruc-residue-graph-set-id
        other-database {
          db "Standard residue dictionary" ,
          tag
          id 1 } ,
        residue-graph-id 61 } } ,
    {
      id 8 ,
      name " 8 " ,
      residue-graph
      standard {
        biostruc-residue-graph-set-id
        other-database {
          db "Standard residue dictionary" ,
          tag
          id 1 } ,
        residue-graph-id 70 } } ,
    {
      id 9 ,
      name " 9 " ,
      residue-graph
      local 1 } ,
    {
      id 10 ,
      name " 10 " ,
      residue-graph
      standard {
        biostruc-residue-graph-set-id
        other-database {
          db "Standard residue dictionary" ,
          tag
          id 1 } ,
        residue-graph-id 67 } } ,
    {
      id 11 ,
      name " 11 " ,
      residue-graph
      standard {

```

```

        biostruc-residue-graph-set-id
        other-database {
            db "Standard residue dictionary" ,
            tag
            id 1 } ,
        residue-graph-id 64 } } ,
    {
        id 12 ,
        name " 12 " ,
        residue-graph
        standard {
            biostruc-residue-graph-set-id
            other-database {
                db "Standard residue dictionary" ,
                tag
                id 1 } ,
            residue-graph-id 68 } } } ,
    inter-residue-bonds {
        {
            atom-id-1 {
                molecule-id 1 ,
                residue-id 11 ,
                atom-id 9 } ,
            atom-id-2 {
                molecule-id 1 ,
                residue-id 12 ,
                atom-id 1 } } } } ,
    {
        id 2 ,
        descr {
            name "B" ,
            pdb-comment "SEQRES" ,
            molecule-type dna ,
            organism {
                org {
                    taxname "synthetic construct" ,
                    db {
                        {
                            db "taxon" ,
                            tag
                            id 32630 } } ,
                    orgname {
                        name
                        partial {
                            {
                                fixed-level other ,
                                level "species" ,
                                name "synthetic construct" } } ,
                                lineage "artificial sequence" ,
                                gcode 11 ,
                                div "SYN" } } } } ,
                seq-id
                gi 996095 ,
                residue-sequence {
                    {
                        id 9 ,
                        name " 21 " ,
                        residue-graph
                        local 1 } ,
                    {
                        id 10 ,
                        name " 22 " ,
                        residue-graph
                        standard {
                            biostruc-residue-graph-set-id
                            other-database {
                                db "Standard residue dictionary" ,
                                tag
                                id 1 } ,
                            residue-graph-id 67 } } ,
                    {
                        id 12 ,

```

```

        name " 24 " ,
        residue-graph
        standard {
            biostruc-residue-graph-set-id
            other-database {
                db "Standard residue dictionary" ,
                tag
                id 1 } ,
            residue-graph-id 68 } } } ,
    inter-residue-bonds {
        {
            atom-id-1 {
                molecule-id 2 ,
                residue-id 1 ,
                atom-id 6 } ,
            atom-id-2 {
                molecule-id 2 ,
                residue-id 2 ,
                atom-id 1 } } ,
        {
            atom-id-1 {
                molecule-id 2 ,
                residue-id 11 ,
                atom-id 9 } ,
            atom-id-2 {
                molecule-id 2 ,
                residue-id 12 ,
                atom-id 1 } } } } ,
    {
        id 3 ,
        descr {
            name "1" ,
            molecule-type other-nonpolymer } ,
        residue-sequence {
            {
                id 1 ,
                name " 9 " ,
                residue-graph
                local 2 } } } ,
    {
        id 4 ,
        descr {
            name "2" ,
            molecule-type other-nonpolymer } ,
        residue-sequence {
            {
                id 1 ,
                name " 21 " ,
                residue-graph
                local 2 } } } ,
    {
        id 38 ,
        descr {
            name "3" ,
            molecule-type solvent } ,
        residue-sequence {
            {
                id 1 ,
                name " 58 " ,
                residue-graph
                local 5 } } } ,
    {
        id 39 ,
        descr {
            name "3" ,
            molecule-type solvent } ,
        residue-sequence {
            {
                id 1 ,
                name " 59 " ,
                residue-graph
                local 5 } } } } ,

```



```

inter-molecule-bonds {
  {
    atom-id-1 {
      molecule-id 1 ,
      residue-id 9 ,
      atom-id 18 } ,
    atom-id-2 {
      molecule-id 3 ,
      residue-id 1 ,
      atom-id 1 } } ,
  {
    atom-id-1 {
      molecule-id 2 ,
      residue-id 9 ,
      atom-id 18 } ,
    atom-id-2 {
      molecule-id 4 ,
      residue-id 1 ,
      atom-id 1 } } } ,
residue-graphs {
  {
    id 1 ,
    descr {
      name " +C DNA" ,
      pdb-comment "" } ,
    residue-type deoxyribonucleotide ,
    iupac-code {
      "N" } ,
    atoms {
      {
        id 1 ,
        name " P " ,
        iupac-code {
          " P " } ,
        element p } ,
      {
        id 18 ,
        name " C5 " ,
        iupac-code {
          " C5 " } ,
        element c } ,
      {
        id 19 ,
        name " C6 " ,
        iupac-code {
          " C6 " } ,
        element c } } } ,
    bonds {
      {
        atom-id-1 16 ,
        atom-id-2 17 ,
        bond-order unknown } ,
      {
        atom-id-1 16 ,
        atom-id-2 18 ,
        bond-order unknown } } } ,
  {
    id 2 ,
    descr {
      name " BR" ,
      pdb-comment "Bromine" } ,
    residue-type other ,
    iupac-code {
      "X" } ,
    atoms {
      {
        id 1 ,
        name "BR " ,
        iupac-code {
          "BR " } ,
        element br } } ,
    bonds {

```

```

    } } ,
  {
    id 3 ,
    descr {
      name " NT" ,
      pdb-comment "Netropsin" } ,
    residue-type other ,
    iupac-code {
      "X" } ,
    atoms {
      {
        id 1 ,
        name " C1 " ,
        iupac-code {
          " C1 " } ,
        element c } ,
      {
        id 5 ,
        name " C2 " ,
        iupac-code {
          " C2 " } ,
        element c } ,
      {
        id 31 ,
        name " N10" ,
        iupac-code {
          " N10" } ,
        element n } } ,
    bonds {
      {
        atom-id-1 1 ,
        atom-id-2 2 ,
        bond-order unknown } ,
      {
        atom-id-1 1 ,
        atom-id-2 3 ,
        bond-order unknown } ,
      {
        atom-id-1 12 ,
        atom-id-2 14 ,
        bond-order unknown } ,
      {
        atom-id-1 29 ,
        atom-id-2 31 ,
        bond-order unknown } } } ,
  {
    id 4 ,
    descr {
      name "MO3" ,
      pdb-comment "Magnesium Ion, 3 Waters Coordinated" } ,
    residue-type other ,
    iupac-code {
      "X" } ,
    atoms {
      {
        id 1 ,
        name "MG " ,
        iupac-code {
          "MG " } ,
        element mg } ,
      {
        id 2 ,
        name " O1 " ,
        iupac-code {
          " O1 " } ,
        element o } ,
      {
        id 3 ,
        name " O2 " ,
        iupac-code {
          " O2 " } ,
        element o } ,

```

```

        {
            id 4 ,
            name " O3 " ,
            iupac-code {
                " O3 " } ,
            element o } } ,
        bonds {
            {
                atom-id-1 1 ,
                atom-id-2 2 ,
                bond-order unknown } ,
            {
                atom-id-1 1 ,
                atom-id-2 3 ,
                bond-order unknown } ,
            {
                atom-id-1 1 ,
                atom-id-2 4 ,
                bond-order unknown } } } ,
    {
        id 5 ,
        descr {
            name "HOH" ,
            pdb-comment "" } ,
        residue-type other ,
        iupac-code {
            "X" } ,
        atoms {
            {
                id 1 ,
                name " O " ,
                iupac-code {
                    " O " } ,
                element o } } ,
            bonds {
                } } } } ,
    model {
        {
            id 3 ,
            type pdb-model ,
            descr {
                name "Model 1 from PDB entry 101D" ,
                pdb-reso "Resolution: 2.25" ,
                pdb-method "X-Ray Diffraction" ,
                pdb-comment "FEB 27 95 Initial Entry" } ,
            model-space {
                coordinate-units angstroms ,
                thermal-factor-units b } ,
            model-coordinates {
                {
                    id 1 ,
                    coordinates
                    literal
                    atomic {
                        number-of-points 556 ,
                        atoms {
                            number-of-ptrs 556 ,
                            molecule-ids {
                                1 ,
                                1 ,
                                1 ,
                                1 ,
                                2 ,
                                2 ,
                                2 ,
                                3 ,
                                4 ,
                                5 ,
                                5 ,
                                5 ,
                                5 ,
                                6 ,

```

```
6 ,  
6 ,  
6 ,  
7 ,  
8 ,  
9 ,  
10 ,  
11 ,  
12 ,  
13 ,  
14 ,  
15 ,  
16 ,  
17 ,  
18 ,  
19 ,  
20 ,  
12 ,  
12 ,  
12 ,  
12 ,  
12 ,  
1 ,  
1 ,  
1 } } ,  
sites {  
    scale-factor 1000 ,  
    x {  
        18598 ,  
        19853 ,  
        20375 ,  
        16812 } ,  
    y {  
        34469 ,  
        34632 ,  
        33233 ,  
        30694 } ,  
    z {  
        24672 ,  
        22605 ,  
        8518 ,  
        -2033 ,  
        26343 } } ,  
temperature-factors  
isotropic {  
    scale-factor 1000 ,  
    b {  
        23239 ,  
        24930 } } } } }
```

**GO background material**

Please see: <http://www.geneontology.org/doc/GO.doc.html>