The SeqHound Manual

Part I: Sections 1-3

For Users

Release 3.3

(April 20th, 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, Christopher Hogue

Edited by
Rachel Farrall and Ian Donaldson

Table of Contents

About this manual. *3*
Conventions *4*
How to contact us. *4*
Who is SeqHound? *5*
1. What is SeqHound? 6
2. How to get started 7
3. Using the SeqHound remote API 8
How to get started with the remote API. *8*
Notes on use of the SeqHound API. *10*
Using the remote API for C *11*
Using the remote API for C++ *12*
Using the remote API for Java *16*
Special Notes 16
System Requirements 19
Using the pre-compiled binary distribution. 20
Test and verify your binary SeqHound Java API installation. 21
Compile, Build and Install Seqhound Java 22
Using the SeqHound Java API 25
API Documentation 26
Configuration 27
Bugs 29
Using the remote API for PERL. *30*
Troubleshooting 32
Documentation 32
Bugs 33
Using the remote API for Bioperl *34*
Prerequisites 34
Installation 34
Documentation 35
Bugs 36
* not available at time of writing

The SeqHound Manual continues with Part II: Sections 4-7. (For Administrators and Developers)

About this manual.

This manual contains everything that has been documented about SeqHound. It is distributed in two Parts (Part I: For Users and Part II: For Administrators and Developers).
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*.
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

The SeqHound Manual (Part I: Sections 1-3) For Users.
Section1 and 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.

The SeqHound Manual (Part II: Sections 4-7) For Administrators and Developers
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.

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 by going to *http://lists.blueprint.org/mailman/listinfo/seqhound.usergroup* .

Private enquiries, bug reports from external users, questions about SeqHound or errors found in this manual may be sent 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 Cavero
Ivy Lu
Rong Yao
Volodya Grytsan
Zhe Wang
Victor Gu
Rachel Farrall
Michael Matan
Elizabeth Burgess
Kai Zheng

1. What is SeqHound?

SeqHound is a bioinformatics programming platform offering daily-updated contents of
1) Entrez and Swiss-Prot sequences,
2) sequence redundancies and neighbours,
3) PubMed and OMIM links to sequence data
4) 3-D structures (NCBI's Molecular Modeling Database),
5) NCBI's taxonomy database,
6) NCBI's complete genomes collection,
7) Entrez Gene contents,
8) protein conserved-domains pre-calculated using RPS-BLAST,
8) Gene Ontology annotations,
9) database cross-references from over 30 databases,

SeqHound is accessible via a remote Application Programming Interface (API) available in Perl, BioPerl, Java, and C/C++. This allows software developers to access SeqHound data from anywhere in the world without having to maintain a local instance of the database or deal with ensuring that the data is up-to-date. Results of daily update scripts and daily API unit tests are available on the SeqHound web-site.
SeqHound may also be set up locally using a freely available user manual and software. SeqHound is ODBC compliant and employs the MySQL database engine. Source code is freely available under the GNU public license (*http://www.blueprint.org/seqhound/*). Larger, pre-calculated data sets are freely available from the SeqHound ftp site.

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 180 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 *to our ftp site at ftp://ftp.blueprint.org/pub/SeqHound/* .

External developers who are interested in developing SeqHound should contact *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.

Open up the SeqHound Resources page when following these instructions.
http://www.blueprint.org/seqhound/seqhound_documentation.html .

1. Read the brief outline of the SeqHound API (see "List of API functions at *http://www.blueprint.org/seqhound/apifunctslist.html* ") to determine if the API will be useful in helping you solve your problem. This section will help you narrow down the set of functions that might be most useful to you (there are over 180 functions in the SeqHound API). For example, you might be most interested in functions listed under GenBank ID Conversions.

   This overview page is also hyperlinked to detailed descriptions for each of the individual functions. The detailed description of the API is maintained at *http://www.blueprint.org/seqhound/api_help/apifunctslist.html* . Documentation for Java versions of these functions can be linked to from this page

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*.

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

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

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

   Also note that "**`Shound`**`GBAccFromGi`" 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.

   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 refer to the link "Function Tacker" which summarizes which API calls are available in which language (see *http://www.blueprint.org/seqhound/apifunctsstatus.html*).

   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.

5. Come back and read the "Notes on use of the SeqHound API" below.

6. The resources page contains all of the links mentioned above. This page (http://www.blueprint.org/seqhound/seqhound_documentation.html) contains other useful links to tutorials and additional information on the API. You may also find a link to daily unit tests for each of the API functions and a link to the update status of SeqHound data sets.

Notes on use 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 "`SHoundGiFromDbNameAndId`".

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 `ShoundGetFasta or SHoundGetFastaKeyedList` rather than ShoundGetFastaList. **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 in our code distribution under *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.

Using the remote API for C

Last updated March 12, 2005.
This section is maintained by Victor Gu.

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 librararies 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).
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 (./makeexmample.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)

Using the remote API for C++

Last updated: April 12, 2005.
This section is maintained by Victor Gu.

To build the C++ version of the remote SeqHound API, you will need the following libraries:

1. NCBI C++ toolkit.
2. SLRI C++ library.
3. SeqHound C++ remote library.

Building the NCBI C++ toolkit library

1. Download the toolkit ncbi_cxx—Nov_30_2004.tar.gz from the NCBI FTP site: ftp://ftp.ncbi.nlm.nih.gov/toolbox/ncbi_tools++/2004/Nov_30_2004/.
   (Note: We noticed the DBAPI package did not compile on Linux Fedora Core 2.0.)

2. Unpack the tar file
   ```
   gunzip ncbi_cxx–Nov_30_2004.tar.gz
   tar ncbi_cxx–Nov_30_2004.tar
   ```

3. Run the autoconf script

   ```
   cd ncbi_cxx–Nov_30_2004
   ./configure
   ```

   (Note: use ./configure –help for different configuration opitons. Default option is –with-debug –without-optimization)

4. Compile the toolkit.

   ```
   cd  ncbi_cxx–Nov_30_2004/GCCxxx-Debug/build
   make all_r
   ```

   (Note: On Intelx86 Linux with RedHat Fedora Core 2.0, the DBAPI package may fail to compile. To skip building the DBAPI package, first remove the target “dbapi” for the project “OPTIONAL_PROJ” in the makefile ncbi_cxx—Nov_30_2004/GCC332-Debug/build/Makefile and then run the above make command.)

Build SeqHound libraries. We assume you are using the UNIX bash SHELL. We also assume you have downloaded and unpacked the SeqHound source tar file as described in section 4.3.

1. Add two environment variables to your login profile .bashrc:

```
export NCBICXX=/path to your home directory/ncbi_cxx—Nov_30_2004
export SLRI=/path_to_your_home_directory/slri
```

2. Save the change to .bashrc and open a new console to source the new environment variables to the SHELL by the following command:

```
source ~/.bashrc
```

3. Edit the following shell script files in the directory slri/lib_cxx/scripts:
   `generate_slrilibstruc.sh`
   `generate_slrilinkset.sh`

   Change the following variables as follows:

```
TOOLDIR=$NCBICXX/GCCxxx/bin/datatool
D=$NCBICXX/src/objects
INTERNAL=$SLRI/lib/asn
```

4. Run the two scripts in the directory slri/lib_cxx/scripts:

```
./generate_slrilibstruc.sh
./generate_slrilinkset.sh
```

5. Modify the make file slri_cxx.mk in the directory slri/lib_cxx/src:

```
BASEDIR=-I$(NCBICXX)/include
```

6. Modify the file Makefile.slri_cxx in the directory slri/lib_cxx/src:

```
builddir=$(NCBICXX)/GCCxxx/build
```

7. Now build the slri library in the directory slri/lib_cxx/src

```
make –f Makefile.slri_cxx
```

8. Modify the files generate_nblastasn.sh and generate_slristruc.sh in the directory slri/seqhound/asn to make sure:

```
TOOLDIR=$NCBICXX/GCCxxx/bin/datatool
D=$NCBICXX/src/objects
```

```
INTERNAL=$SLRI/seqhound/asn
```

9. Run the two scripts in step 8 the same way as in step 4.

10. Edit the make file seqhoundrem_cxx.mk in the directory slri/seqhound/src_cxx to ensure:

```
BASEDIR=-I$(NCBICXX)/include
```

11. Edit the file Makefile.seqhoundrem_cxx in the directory slri/seqhound/src_cxx to ensure:

```
builddir=$(NCBICXX)/GCCxxx/build
```

12. Make the seqhound remote API library in the same directory as in step 11:

```
make –f Makefile.seqhoundrem_cxx
```

13. To build a test application that uses the SeqHound C++ remote library, edit the make file Makefile.mytest in the directory slri/seqhound/src_cxx to ensure:

```
builddir = $(NCBICXX)/GCCxxx/build
CXXINDIR = -I$(NCBICXX)/include -I$(NCBICXX)/GCCxxx/inc
INCOBJDIR = -I$(NCBICXX)/include/objects
```

14. Create the test application in the directory slri/seqhound/src_cxx by the following command:

```
make –f Makefile.mytest
```

15. Edit the seqhound http server and cgi path configuration file shound.ini in the directory slri/seqhound/src_cxx

```
server1 = seqhound.blueprint.org
(This is the default, you can switch to the URL pointing to
your local seqhound installation(e.g.127.0.0.0))
CGI=/cgi-bin/seqrem
```

16. Run the C++ test program mytest in the same directory as in step 15:

```
./mytest
```

17. The test program will log the test result to a file named sample_test.txt2 in the same directory as in step 16.

Using the remote API for Java

Last updated April 14, 2005
This section is maintained by Michael Matan.

NOTICE TO FIRST TIME USERS OF THE JAVA API.
You may skip the 'Special Notes' section below and go directly to the System Requirements section.

NOTICE TO THOSE CURRENTLY USING VERSION 3.2 OF THE JAVA API

1) As of release 3.3, the contents of the *shconfig.properties* file has changed to point to a SOAP service that supports new API functions. You must update to the latest version of this file or include the line:

`SOAPBaseURL=`*http://seqhound.blueprint.org:8080/soap/services*

2) As of release 3.3, the function signatures of the API functions in the GenBankIDConversion Interface have changed. Please refer to the JavaDocs for this interface and make the appropriate changes to your code. These changes have been made to better comply with the Jax-RPC standard and to guarantee interoperability between the underlying SOAP service and other languages.

NOTICE TO USERS OF THE 3.01 (OR EARLIER) VERSION OF THE 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 existing applications using the SeqHound Java API before upgrading to v3.2 and later. Version 3.01 (and earlier) of the Java API will be supported for a limited time so 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:
1. 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., implentations 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.

2. 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.

3. 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:

4. 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.
5. 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.
6. The method SHoundGetFastaList(int[] giList, Writer out) has been renamed ShoundGetFastaListToWriter

7. 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 neccesitated 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);
```

8. 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

9. Functions listed on the Function Tracker page (*http://www.blueprint.org/seqhound/apifunctsstatus.html*.) as having a Java Local version are only supported by versions 3.3 (and higher) of the Java remote API.

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 3rd 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 "Configurating 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. The class is org.blueprint.seqhound.SeqhoundAPITest, and can be invoked through either an ant task or directly through the Java interpreter.

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:lib/axis/axis.jar:lib/axis/commons-
discovery.jar:lib/axis/jax-rpc.jar:lib/axis/saaj.jar:lib/axis/wsdl4j.jar


java org.blueprint.seqhound.SeqhoundAPITest
```

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 only those class files
#required for the seqhound remote distribution.
ant dist-remote

#or if you have your own local installtion of SeqHound
#to create a jar containing all the seqhound local and
#remote client library class files
ant dist-local
```

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)
n.) *java/soap* (contains the SOAP related source files)
o.) *java/autogensrc* (contains autogenerated source files)

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/

`ws-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/

`deploy-ws` - deploys the (WAR) file built by the dist-war target to a running tomcat server, as specified in the conf/tomcat.properties file

`undeploy-ws` - undeploys the web application deployed by the tomcat-deploy target

`redeploy-ws` - undeploys and redeploys the jseqrem servlet web application; synonymous with calling 'undeploy-ws' and 'deploy-ws' in sequence.

`test` - runs the junit test for org.blueprnt.seqhound.SeqhoundAPITest 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 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 is a class for verifying that your seqhound build is working properly.
This test class was built using the junit framework, and iterates through nearly all seqhound API functions to verify that they are working correctly in your configuration. The class is org.blueprint.seqhound.SeqhoundAPITest .

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, place the jars in the lib/ and lib/axis/ directories in your java CLASSPATH and then enter on a single line:

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

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 use of the seqhound method 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:

1. log4j settings, for configuring logging
2. What datasources to use in the API implementations
3. Method-specific data source overrides.
4. 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 emmitted by
loggers in the org.blueprint.seqhound sub-tree
log4j.logger.org.blueprint.seqhound=INFO
```

To learn more about log4j and it's 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 functions that have a Local Java version (see the SeqHound function tracker at *http://www.blueprint.org/seqhound/apifunctsstatus.html*. 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.

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 invokations of the API method SHoundGBAccFromGi should utilize direct local seqhound database access to retrieve the result. The line:

ShoundGiFromGBAcc=remote

specifies that invokations 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 seqhound SOAP servers and local database servers.
Remote seqhound SOAP servers are specified with the *SOAPBaseURL* property. This should be set to the URL of the directory containing the seqhound SOAP services to be

accessed by the client. To set this value to the web directory containing the Seqhound project's public SOAP services, enter the following line in your configuration file:

```
SOAPBaseURL=http://seqhound.blueprint.org:8080/soap/services
```

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

```
cseqremURL=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 explicitely 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* 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*

   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/slritools
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* 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* 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* 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

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* with details.