

Version

1.0.2

CANADA'S MICHAEL SMITH GENOME SCIENCES CENTRE

CHINOOK USER GUIDE

GENOME SCIENCES CENTRE, BCCA

Chinook User Guide

© Genome Sciences Centre
BC Cancer Agency
Suite 100
570 West 7th Ave
Vancouver, BC
V5Z 4S6
Phone: (604) 707-5800
First Floor Fax: (604) 876-3561
Fifth Floor Fax: (604) 733-9481

Table of Contents

Table of Contents	i
1.0 Introduction.....	1
1.0.1 Feature Matrix	2
1.0.2 Background / Use Cases	3
2.0 Installation.....	4
2.0.1 Release builds	4
2.0.2 CVS	4
3.0 Running Chinook.....	5
3.0.1 Running the Client.....	5
3.0.1.1 Using the Installers	5
3.0.1.2 From Webstart	5
3.0.1.3 Using ANT	5
3.0.1.4 Using the Code	7
3.0.2 Running the Server.....	11
3.0.3 P2P	12
3.0.3.1 Running the P2P process from the Installer.....	12
3.0.3.2 Running the P2P process within the Chinook Client.....	12
3.0.3.3 Running the P2P process from ANT	13
3.0.3.4 Deconstructing the P2P Networking abilities of Chinook.....	13
3.0.3.1.1 JXTA	13
3.0.3.1.2 Advertisements.....	13
3.0.4 PERL.....	15
3.0.4.1 Setting up the Chinook Perl Environment	15
3.0.4.2 Starting the Perl Engine (Client).....	16
3.0.4.3 Running PERL Scripts.....	17
4.0 Customizing Chinook	18
4.0.1 Using the Resources directory	18
4.0.1.1 Configuring static services	18
4.0.1.2 Adding services	20
4.0.1.3 Configure server information.....	20
5.0 Walkthroughs.....	22
5.0.1 Running a job with Chinook (using the GUI)	22
5.0.2 Adding a new Service	30
5.0.3 Setting up a Chinook server node.....	33
5.0.4 Running a batch Perl job	38
6.0 Further Information	43
6.0.1 Mailing List	43
6.0.2 Authors	43

6.0.3 Known Problems.....	44
6.0.4 License.....	44

1.0 Introduction

Chinook is a peer-to-peer (P2P) bioinformatics platform. The goal of the Chinook platform is to facilitate the exchange of analysis techniques and resources within a local community and/or worldwide. Chinook operates by turning command-line applications/utilities into services which are advertised over a virtual network. Currently, there are multiple analysis services that have been made accessible by Chinook. These range from alignment to regulation prediction algorithms. Furthermore, Chinook adding new services is facilitated using XML (A GUI is under development to facilitate service configuration when this manual is written).

Chinook clients can be operated from Java, Perl, or within applications like [Sockeye](#) (And soon [Pegasys](#) at the Ouellette Lab in Vancouver, and OrthoSeq project at the [Wasserman Lab](#) in Vancouver (CMMI)).

IS THIS MANUAL FOR YOU?

This manual is for users who are interested in running Chinook. Chinook can be run in several different ways:

1) As a Client. This is for users who are interested in running various command-line tools. These users generally have an idea of where Chinook servers are located on a network and are interested in interacting directly with these servers. The locations of servers can be specified by adding a static service description in XML or adding the location from the Tools menu of the Chinook Client User Interface.

2) As a Server. This is for users who want to turn command-line applications into services that can be accessed by a Chinook client. This can be done using either Web Services or Java Remote Method Invocation (RMI).

3) As a Client using Peer-to-peer. This is for users who want to discover the location of servers and their corresponding services on their network. They generally do not know what servers/services are available and are interested in discovering this information.

4) As a Server using Peer-to-peer. This is for server providers who want to enable users to discover their server over a network. They will start a peer-to-peer process and then subsequent servers will be registered and advertised to the network from this process (servers communicate with the peer-to-peer process using TCP/IP sockets). This could generally be an individual with a new tool who wants to advertise their tools existence to other Chinook users.

5) As a Perl-Engine Client. A Chinook client will be started with the batch (perl engine) activated. This will allow users to write scripts using the Chinook perl modules

to discover and process jobs over the Chinook network. This is for users who want to distribute a large number of jobs to multiple locations. The Perl Engine can be run in either a peer-to-peer mode (if the peer-to-peer process was initiated) or in a client-server mode (if a list of static services has been written to the appropriate XML file).

TERMINOLOGY:

RMI: Java Remote Method Invocation is a Java-to-Java communication protocol for transferring Java objects over a network or instance of a Java Virtual Machine.

Web Services: Web services are language-independent communication protocol utilizing SOAP (Simple Object Access Protocol). SOAP is responsible for describing various objects in XML representations.

1.0.1 Feature Matrix

Server	Integrates command-line applications in XML	✓
	Allow multiple input/output files	✓
	STDERR/STDOUT Previewing	✓
	RMI (remote method invocation)	✓
	WSDL (web services)	✓
	Ant startup	✓
	Add new services (GUI supported)	✓
	Add static services (GUI supported)	✓
	Provide access for server-side databases (i.e. Ensembl)	✓
	File Storage memory allocation is user configurable	✓
Client	Run and kill services (GUI supported)	✓
	Run batch files	✓
	Download results	✓
	Specify filters to narrow search	✓
	Allow preview of service webpage	✓
	Contain server information dialogs (GUI supported)	✓

P2P	Discover nodes	✓
	Advertise services	✓
	Multiple clients and servers supported	✓
	Auto-configuration supported for JXTA	✓
Perl	Run batch job through command-line	✓
	Submit batch jobs to queue	✓
	Monitor service and job information	✓
	Download results	✓

[Back to Table of Contents](#)

1.0.2 Background / Use Cases

Bioinformatics techniques are used to identify complex, re-occurring relationships in biological data. Genome sequencing projects and high-throughput expression analyses have contributed large amounts of data; both complicating analysis and demanding higher-level coordination of computational resources. Furthermore, the variety of available bioinformatics tools and algorithms, and their diverse modes of usage create a situation where most users have trouble discerning where to invest their time and resources. Chinook resolves these issues by creating a virtual network for bioinformatics analyses. A user is able to dynamically resolve available bioinformatics services (algorithms) over the Internet or their local network. The user can then validate a server's authenticity and submit bioinformatics analyses to peers that publish their ability to perform desired services. Information like bandwidth, jobs in queue and the location of Chinook services are reported to clients to aid in their job submission process. A user is also able to visit the service creator's website to identify what the particular service does. Chinook allows a service provider to create a new service by simply editing an XML file; as long as the new service has a standard output format, no additional programming is required. The Chinook server runs over the JXTA peer-to-peer network in both Java Remote Method Invocation (RMI) mode or through Apache Axis web services. Chinook creates a virtual community where researchers can rapidly hone in on applications of interest to run them across multiple service providers while shifting the responsibility for application maintenance from the client to the developer.

[Back to Table of Contents](#)

2.0 Installation

There are two ways to install Chinook; you can download a tarred and zipped release, or checkout the source from CVS. This section will describe how to do each of these tasks. For detailed information on setting up a server see the Walkthrough in Section 5.

2.0.1 Release builds

Chinook versions are available as tarred and zipped files from <http://smweb.bcgsc.bc.ca/chinook/download.html>. To run Chinook, you will need to download one of these files, unzip it, and run the desired process using Apache Ant.

Apache Ant is a make utility that is available from <http://ant.apache.org>. It is our goal to include executable jars in the near future as well to ease the running of services.

2.0.2 CVS

The Chinook code is available over CVS from Java.net as part of the Global Learning and Education project space. (NOTE: For more information on what CVS is, read <https://www.cvshome.org/docs/manual/>.)

To login to the CVS repository type in the following command.

```
cvsc -d :pserver:USERNAME@cvsc.dev.java.net:/cvsc login
```

Replace the USERNAME with your user name, and type in your Java.net password when being prompted. Then type in the following command to checkout Chinook.

```
cvsc -d :pserver:USERNAME@cvsc.dev.java.net:/cvsc checkout chinook
```

Now you have downloaded a working copy of Chinook. You can use Apache Ant (<http://ant.apache.org>) to run various process with your copy of the Chinook code.

For more information on how to connect to CVS from a GUI or even from an IDE like Eclipse, please refer to **Chinook Developer Guide**.

[Back to Table of Contents](#)

3.0 Running Chinook

There are several different ways you can run Chinook (see the Introduction for more information). This section describes each of the different ways that Chinook can be run: as a client, as a server, as a P2P-enabled client or server, or as a client running the Chinook Perl Engine (capable of submitting jobs from batch scripts).

3.0.1 Running the Client

There are several different ways you can run a Chinook Client. The first two methods below are recommended for all users. Using ANT or running from the source code is recommended for more computer-savvy individuals.

3.0.1.1 Using the Installers

From the Chinook download page (<http://www.bcgsc.bc.ca/chinook>) you can download installers for Windows, Linux, and Mac OS X. Just download one of these installers and run it, it will guide you through the procedures of installation.

Once installed, you will be able to run the “**Chinook Client**” from the “**Chinook P2P Bioinformatics**” program group. Additionally, you can start-up the peer-to-peer services or the Perl engine (for writing batch scripts).

3.0.1.2 From Webstart

You can run the Chinook Client from Java Web Start by running Chinook from the web start website from, <http://www.bcgsc.ca/chinook>. Using Java Web Start, allows you to keep in sync with latest versions of Chinook without requiring the download of new installers (Web Start will dynamically update the components which are out of date).

When at the BCGSC Chinook website, click **Latest Download** from the top-right column, and then click the **Chinook Client Web Start** link to open the web page containing the Chinook Client Java Web Start link. Click on the corresponding link to start the Chinook Client.

3.0.1.3 Using ANT

Ant is a Java-based build tool developed by the Apache Software Foundation (<http://ant.apache.org>). Once Ant is installed, the Client can be started by running:

```
ant client
```

Ant will use the default **build.xml** file in the installation directory and run the target **client**. Ant is extremely powerful tool since it will automatically configure and build the Chinook code so that it works on your system.

3.0.1.4 Using the Code

If you are interested in customizing Chinook for your own purposes, you can download the Chinook code and modify it. Running a modified Chinook client requires using the ant task mentioned above. This section discusses how to set-up Chinook to run from the source code or in an IDE like Eclipse.

Note: If not using ANT, ensure all the dependency jars in the **chinook/lib/** folder are added to the CLASSPATH before you compile any Chinook code. Furthermore, it is required that the **chinook/resources/** folder is also added to the CLASSPATH. The **resources** folder contains all the user-configurable Chinook files.

There are several options when running the Chinook Client from the code. The main method of the Chinook Client is located in **ca.bcgsc.chinook.client.exec.ChinookClient.java**. Chinook can be started with either the GUI hidden or in batch mode; these modes are specified with the application arguments:

- nogui** Turn off the GUI
- batch** Enable the PERL Engine (embedded in client)

Since the peer-to-peer (P2P) connectivity of Chinook is handled through a separate process, you may want to also run **ChinookP2PNode** first; there are no application arguments for the **ChinookP2PNode**. The main method of the **ChinookP2PNode** is located at **ca.bcgsc.chinook.p2p.exec.ChinookP2PNode.java**. For more about the P2P functionality in Chinook see section 3.3.

The default execution mode of the Chinook Client is to display the GUI and to not allow batching through the Perl Engine.

Chinook for Eclipse

In the rest of this section, we will walkthrough setting up Chinook in an Eclipse environment.

Depending on which IDE you are using to customizing Chinook project, the procedure may be different. For example, in Eclipse you can go to **Run** menu, then click **Run...** menu item. The window in Figure 3.1 should appear.

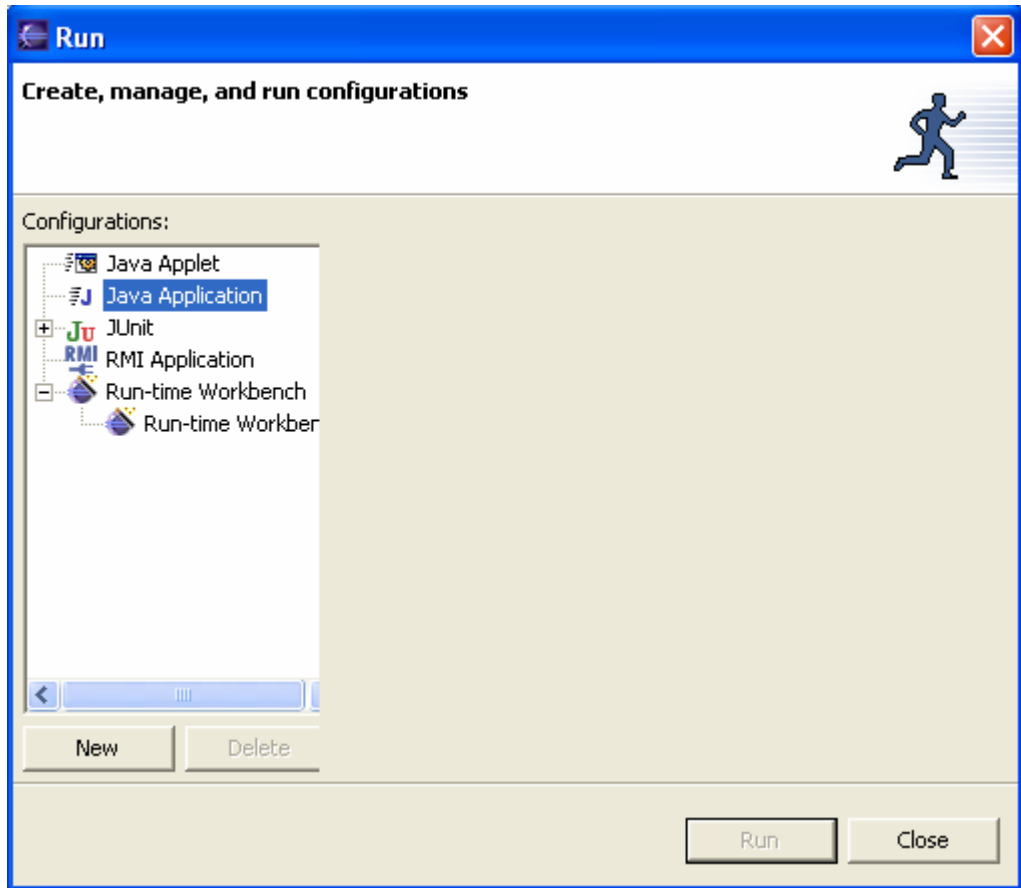


FIGURE 3.1 Run window. Select which main method to run.

Click **Java Application** in the **Configuration** window then click the **New** button under it. In the **Name** text field, type in “ChinookP2PNode”. If there is something in the text field besides the Search button, clear it, and then click the **Search** button and the following window will appear.

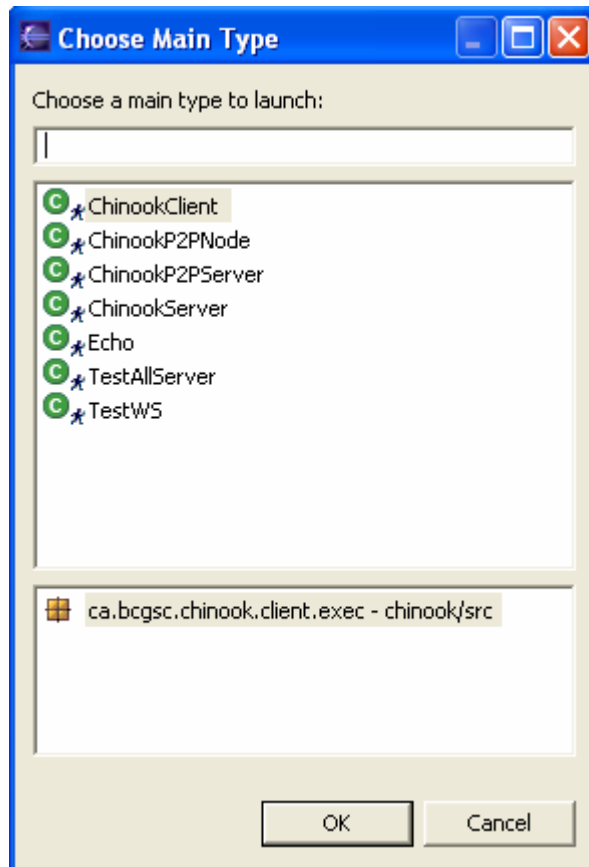


FIGURE 3.2 Choose Main Type Window. It displays all the main methods available.

Select the **ChinookP2PNode** and then click the **OK** button. We are going back to the **Run** window. Click the **Apply** button, and then the **Run** button. The **ChinookP2PNode** will start.

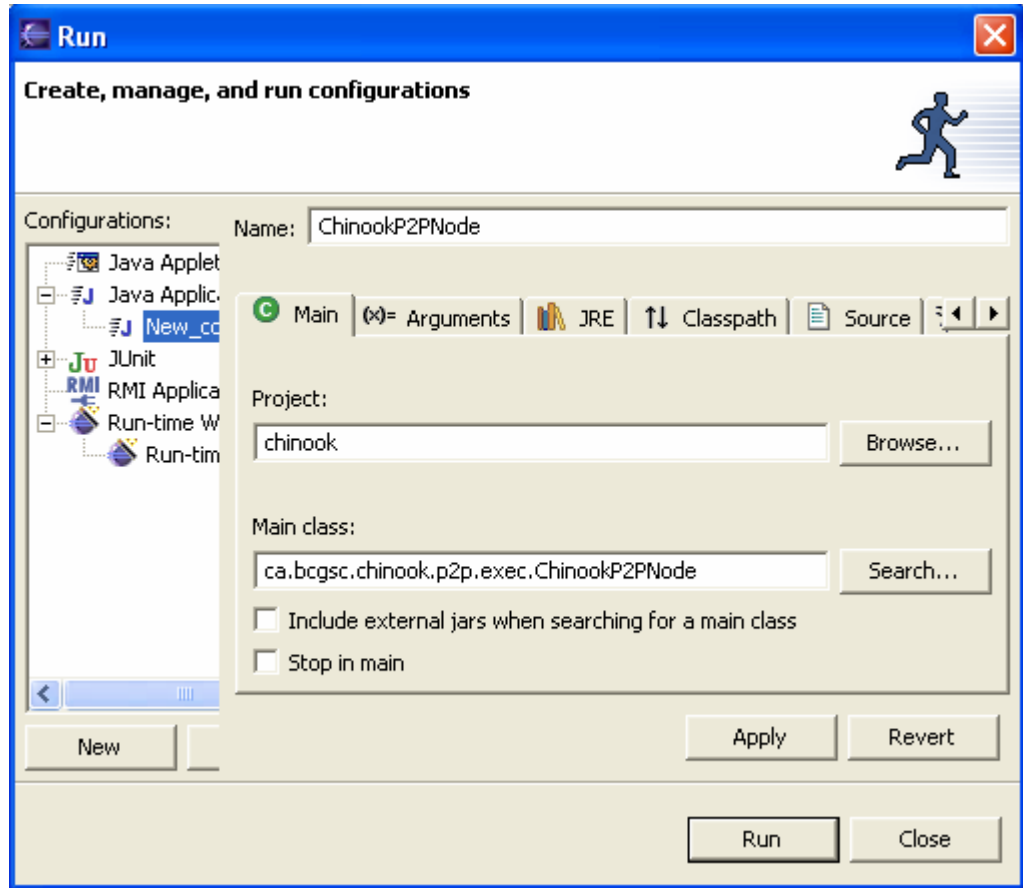


FIGURE 3.3 Run window. Running ChinookP2PNode.

Follow the same procedure to run the Chinook Client. The difference is to select **ChinookClient** from the **Choose Main Type** window.

[Back to Table of Contents](#)

3.0.2 Running the Server

Currently, running the Chinook Server is a bit more involved than running the Client. This is because it requires a user to choose how they want to deploy services (either through Tomcat/Axis or Java RMI) and install the appropriate dependencies. Additionally, it requires a user to configure their services, deploy them to Tomcat (if necessary), and ensure that they are advertising their server correctly on the P2P network.

As mentioned above, there are two ways you can deploy services to a client. One is through Web services (Tomcat/Axis) and the other is through Java RMI (Remote Method Invocation). If you want to run the server in Web Services mode, you need to install and configure a Tomcat server with Apache Axis. There are no installation dependencies for RMI except a valid Java installation which can be downloaded from <http://java.sun.com> as a JDK or JRE.

To configure the server, it is recommended to read the rest of this section and then go through the Walkthroughs in Section 5 on setting up a server and adding a service.

Assuming that the Chinook server is configured, the server can be run using Ant (which is the preferred mechanism until successive releases bundle it in the Installer, with a server configuration GUI). The command to start the server is:

```
ant server-start
```

To stop the server type:

```
ant server-stop
```

[Back to Table of Contents](#)

3.0.3 P2P

Chinook is designed to use a peer-to-peer (P2P) network to facilitate the discovery and exchange of bioinformatics utilities. The P2P functionality is run as an autonomous service which clients and servers can connect to; for receiving lists of discovered services or requesting that a new server's location be advertised. Regardless of how you want to use the P2P functionality, it is a prerequisite that clients and servers connect to a **previously running** instance of the P2P process. Therefore, if you are planning on using the P2P functionality, you will need to run the P2P process **first** and then any subsequent clients or servers (there is an exemption for the client's using the GUI, who can start the P2P process internally from the Tools menu). The rest of this section describes various ways in which the P2P process can be run and more technical information on how Chinook establishes its P2P network.

3.0.3.1 Running the P2P process from the Installer

From the Chinook download page (<http://www.bcgsc.bc.ca/chinook>) you can download installers for Windows, Linux, and Mac OS X. Just download one of these installers and run it, it will guide you through the procedures of installation.

Once installed, you will be able to run the “**Start P2P Node**” process from the “**Chinook P2P Bioinformatics**” program group. This will start the P2P process in a new shell. You can either close this window to stop the P2P Node or explicitly run “**Stop P2P Node**” from the “**Chinook P2P Bioinformatics**” program group.

3.0.3.2 Running the P2P process within the Chinook Client

It is possible to run the P2P Node from the client explicitly by starting up the Chinook Client GUI and then visiting the “**Tools**” menu and clicking on “**Activate P2P**”. The status of the client's connection to the P2P Network is shown in the lower-right status bar at the bottom of the main application window.

3.0.3.3 Running the P2P process from ANT

Running Chinook in a P2P environment using Ant is similar to running the server using Ant. At the prompt, type:

```
ant p2p-start
```

To stop the P2P Node type:

```
ant p2p-stop
```

3.0.3.4 Deconstructing the P2P Networking abilities of Chinook

This section gives a technical overview of how the Chinook P2P network is constructed and used to ultimately exchange information regarding Chinook service locations over the Internet.

3.0.3.1.1 JXTA

Chinook servers publish advertisements using the JXTA protocol (For more information about JXTA protocol, visit <http://www.jxta.org/>). The client peer intercepts XML advertisements and displays services (for which jobs can subsequently be run). The next section describes how Chinook advertisements are made and how you can edit them for your services.

3.0.3.1.2 ADVERTISEMENTS

An advertisement is an XML document that describes a particular JXTA message, whether that is a peer, peer group or service. These messages are discovered then cached locally. (To see your cache go to your own `.jxta/` directory - created when you first run Chinook). As a server/service provider, you are interested in only two types of service messages, the `ModuleSpecAdvertisement` (MSA) and the `ModuleImplAdvertisement` (MIA). These are located in your `advertisements/` directory. (Chinook automatically handles peer group and peer advertising).

The Chinook `ModuleSpecAdvertisement`

Chinook has two `ModuleSpecAdvertisement`'s in its `advertisements/` folder. One for using RMI and another for using the web services protocol. Depending on how you want to run your server, you will currently need to modify one of these advertisements to point to your servers location. For the purpose of this example, we will look at the RMI advertisement.

`chinookSpecAdvRMI.xml`

```
<?xml version="1.0"?>
<!DOCTYPE jxta:MSA>
<jxta:MSA xmlns:jxta="http://jxta.org">
  <MSID>
    urn:jxta:uuid-
    72CE4F415C994ADBB5ECB897E6BBB3D0EB39B9952C0D4D79BAD5BDE678877
```

```

F4D06
</MSID>
<Name>JXTASPEC:Chinook-RMI</Name>
<Crtr>smontgom@bcgsc.bc.ca</Crtr>
<SURI>http://www.bcgsc.ca/Chinook/</SURI>
<Vers>1.0</Vers>
<Desc>A Chinook DBAS RMI Server</Desc>
</jxta:MSA>

```

The **<MSID>** tag holds a unique id that identifies this service, for your purposes it can be any valid JXTA id (see <http://spec.jxta.org/nonav/v1.0/docbook/JXTAProtocols.html> JXTA Protocol Specification). The **<Name>** tag shouldn't be changed; Chinook searches for Spec advertisements based on this. If the **<Name>** tag is changed, your advertisement won't be discovered. The **<Crtr>** tag specifies who the publisher of this service is (This is automatically configured from your **resources/applications.xml** file and does not need to be changed). The **<SURI>** tag points to the documentation for Chinook; but this can be changed to be any service providers' relevant service documents. The **<Vers>** tag specifies what version of the Chinook is being used (this is also automatically configured). Finally, the **<Desc>** tag describes the service.

The Chinook ModuleImplAdvertisement

Chinook has two **ModuleImplAdvertisement**'s in its **advertisements/** folder. These correspond to specific implementations of the **ModuleSpecAdvertisement**'s. Any service provider **MUST** change these advertisements to reflect the appropriate server information (the URI). We will look at the RMI example of the **ModuleImplAdvertisement**.

chinookImplAdvRMI.xml

```

<?xml version="1.0"?>
<!DOCTYPE jxta:MIA>
<jxta:MIA xmlns:jxta="http://jxta.org">
  <MSID>
urn:jxta:uuid-
72CE4F415C994ADBB5BCB897E6BBB3D0EB39B9952C0D4D79BAD5BDE678877
F4D06
  </MSID>
  <Comp>
    <Efmt> JDK1.4 </Efmt>
    <ChinookImpl> 1.0 </ChinookImpl>
  </Comp>
  <Code>//localhost:1099/ApplicationServerImpl</Code>
  <PURI>Not yet available</PURI>
  <Prov>smontgom@bcgsc.bc.ca</Prov>
  <Desc>RMI Chinook Implementation, Parm is generated with
service names
  </Desc>
</jxta:MIA>

```

The **ModuleImplAdvertisement** here has only a few things that must be noted. The **<MSID>** tag must match that of the corresponding **ModuleSpecAdvertisement**. The **<Comp>** tag specifies compatibility information. Here it states that users must have at least JDK1.4 and the 1.0 implementation of the Chinook interface. The **<Code>** tag points to the relevant URI to run the Chinook service. The **<PURI>** tag specifies a location to download the appropriate classes; this is not yet implemented. The **<Prov>** tag specifies who is providing this implementation. Finally, the **<Desc>** tag describes the implementation. If you have seen a discovered advertisement, you may be wondering why they are different from this description.

A discovered Chinook **ModuleImplAdvertisement** will have automatically filled in **<service_type>** and **<service_name>** elements as part of the **<Param>** tag. The **<Param>** tag specifies additional Chinook metadata. Chinook uses this to ontologically classify services. This allows versions of Chinook to discover advertisements of only the type a client is interested in.

[Back to Table of Contents](#)

3.0.4 PERL

Chinook has been integrated with Perl to allow the automated discovery and execution of services from scripts. The Perl code is compatible with Bioperl (www.bioperl.org). This section describes how to set-up your environment to be able to run Perl scripts for Chinook. For a walkthrough of creating a Perl script for Chinook also see the Walkthrough in [Section 5](#).

3.0.4.1 Setting up the Chinook Perl Environment

Perl needs to know the location of the Chinook Perl modules. The modules for Chinook are installed under the **perl/modules/** directory in the Chinook installation directory. To point Perl at these modules, you can set the **PERL5LIB** environment for your shell. This can be performed by issuing the following commands.

If you are using tcsh/csh shells:

```
setenv PERL5LIB ${PERL5LIB}:${CHINOOK_HOME}/perl/modules
```

In bash, the equivalent command is:

```
export PERL5LIB=${PERL5LIB}:${CHINOOK_HOME}/perl/modules
```

Where **`\${CHINOOK_HOME}`** is the location of your Chinook installation.

NOTE: We typically prefer to write these commands to our user **.bashrc** file in our user directory to prevent having to retype them every time we want to use the Chinook Perl modules.

Alternatively, you can use the perl pragma `'use lib'` at the top of your scripts to point to the location of the Perl modules you wish to use:

```
use lib '/home/chinook_install_directory/perl/modules';
```

Where `/home/chinook_install_directory/` is the installation directory for Chinook.

3.0.4.2 Starting the Perl Engine (Client)

To discover services and run analyses, an instance of the Chinook Client must be running in batch mode with a port open for communication with Perl scripts. The Perl scripts connect to this port to determine what services are available and to send requests for execution.

To set-up the Chinook Client for execution in batch mode (for Perl):

- 1) Open the `batch-config.xml` file in the `resources/` directory in your Chinook installation directory.
- 2) There are several tags that need to be set.
 - a. `<batch_directory>` specifies the directory where information about discovered services is written (batch files). Whenever a new service is discovered, a batch file is written to this directory describing the service, its location, and required parameters for execution.
 - b. `<batch_queue_directory>` specifies the directory where completed batch files are stored (batch_queue files). A complete batch file has parameters and data set and is ready for execution. It also has a batch_queue id attached to it to identify downstream output files. The Chinook Client can be notified to read all the files in this directory and process them. (Alternatively, it can be given the location of a batch_queue file directly)
 - c. `<batch_reporting_directory>` specifies the directory where completed report information is written to (batch_reports). The Chinook Perl code usually polls this directory for reports matching a specific batch_queue file id.
 - d. `<batch_machine_name>` specifies where the Client is executing. Usually localhost is sufficient. In NFS mounted systems, an explicit machine name is required. The Chinook Perl code needs to know this location to be able to connect to the Chinook Client.
 - e. `<batch_port>` specifies the port that the Chinook Client will be receiving incoming requests from Perl scripts. The default port is 7999. This can be any valid open port number, but Perl clients will

need to know this information in addition to the `<batch_machine_name>` in order to connect to the Chinook Client.

- f. `<batch_socket_conns>` specifies the maximum number of open socket connections that can be made to the Chinook Client at a time. This number should usually be greater than `<batch_receiver_thread_queue_size>`.
- g. `<batch_receiver_thread_queue_size>` specifies the maximum number of concurrent processing requests that scripts can make of the client. The rest will block until the pending requests are finished. This should be a low number to prevent excessive use of memory. But should be inline with the number of concurrent requests your Chinook Client receives.

- 3) Once the `batch-config.xml` file has been configured to your desired settings, the Chinook Client can be started with batch-mode enabled. This starts a small server inside the Client that will manage incoming requests from Perl scripts.
- 4) To start the Chinook Client with batching mode, start the Client as normal but with the following argument (or, if using an installer, run the **“Start Perl Engine”** executable from the **“Chinook P2P Bioinformatics”** program group).

`-batch`

The batch flag will ensure that the batching mode is activated. If you do not want the GUI to appear, you can call the Chinook Client with:

`-batch -nogui`

This is ideal for running the Chinook Client on remote machines.

3.0.4.3 Running PERL Scripts

For examples and more information on running Perl scripts once the environment has been configured and batch-enabled client has been started, see the Perl Walkthrough in [Section 5.0.4](#).

[Back to Table of Contents](#)

4.0 Customizing Chinook

In this section, we are going to guide you through how to customize common features of Chinook for your own installation, including configuring static services, adding new services, and configuring server information.

4.0.1 Using the Resources directory

As Chinook starts, it uses xml files under **resources/** directory to configure itself. Many of Chinook's features can be customized by editing the xml files in this directory.

4.0.1.1 Configuring static services

You can add static services to Chinook by editing the **static-services.xml** file under **resources** directory (or add them in the GUI through the “**Tools**” → “**Static Services...**” menu item). Each time Chinook starts; it tries to connect to servers in **static-services.xml** first instead of trying to discover services them through the P2P network. This can be useful if you know the location of the service you want to access or have a private service (for debugging, etc.). Below is an example of the **static-services.xml** file.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<static-services>
  <staticservice>
    <URI>//localhost:1099/ApplicationServerImpl</URI>
    <mode>RMI</mode>
  </staticservice>
  <staticservice>
    <URI>http://localhost:1099/ApplicationServerImpl</URI>
    <mode>WSDL</mode>
  </staticservice>
</static-services>
```

The **<URI>** tag defines the location of the server, and **<mode>** tag defines the mode of the server, either RMI (remote method invocation) or WSDL (web service). You can add more than one static server. Enclose your static between **<staticservice>** and **</staticservice>** tags.

As mentioned, you can also edit the **static-services.xml** file through the GUI. After you start the Chinook Client, click on the **Tools** menu. Click the **Static Services...** menu item. A window similar to the following will appear.

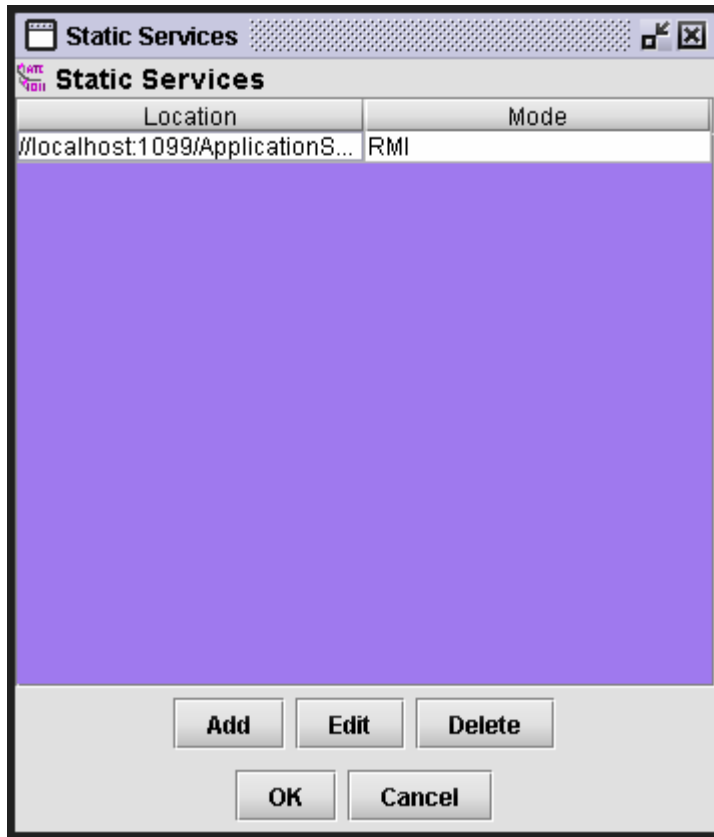


FIGURE 4.1 Static Services window. It displays all static services currently available.

You can add new static services by clicking the **Add** button. After clicking the **Add** button, the following window will appear.



FIGURE 4.2 Static Services Editor window. It is used to add or edit static services.

Type in the server location and choose the type of the service, then click **Test Connection** button. If the client is able to connect to the server, the red square will become green, and the **OK** button will become enabled. If it can not be connected, the red square will remain red, and you cannot add the static service to the **static-services.xml** file.

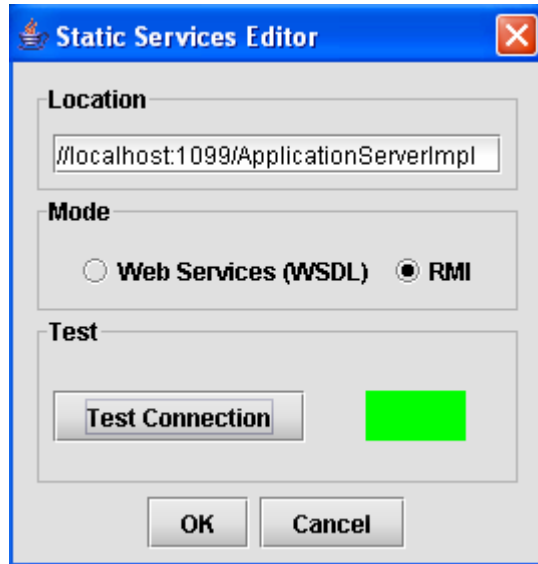


FIGURE 4.3 Static Services Editor window. Use this window to add a new static service.

After you've finished, click the **OK** button on the **Static Services Editor** window, and then click the **OK** button on the **Static Services** window. All the static services you just added will be written to the **static-services.xml** file. Editing existing static service is similar. You can delete a static service by selecting the service you want to delete in the **Static Services** window, and then click the **Delete** button.

4.0.1.2 Adding services

Each time Chinook server starts, it checks all the services specified in **applications.xml** file, which is located under **resources/** directory, to see if their executables exist and then publishes them. In order to add new services to your Chinook Server, you currently need to describe a new service using xml tags (a GUI will soon be available to facilitate this process). For a more detailed example, you can go to [5.0.2 Adding a new Service](#).

4.0.1.3 Configure server information

If you want to run as a Chinook Server, you can also customize the **server-info.xml** file, which is under the **resources/** directory. This allows clients to obtain information about your server. An example of the **server-info.xml** file is below.

```
<server-info>
  <location>Genome Science Centre</location>
```

```
<description>  
  This is the description of the server  
</description>  
<contact>chinook@bcgsc.bc.ca</contact>  
</server-info>
```

The **<location>** tag defines you server location, it could be the URL of your website. The **<description>** tag specifies the description of the server. The **<contact>** tag defines the contact information for the server (typically a maintainers' email address).

[Back to Table of Contents](#)

5.0 Walkthroughs

In this section, we are going to guide you through using several common features available in Chinook, including running a job using a GUI, adding a new service, and running a batched Perl job.

5.0.1 Running a job with Chinook (using the GUI)

In this walkthrough, we will show you how to run a job in Chinook using the graphical user interface.

Step 1: Starting Chinook Client.

After you start the Chinook Client, the following window will appear (See Figure 5.1). There are five main components in Chinook client user interface. The first one is the **Chinook Menubar** (top of the window under the title bar). The second one is **Service Type and Filter Panel** (upper-left panel), which is used to display all the available service types and used to specify filters to narrow the search of the services you want. The third one is the **Discovered Services Panel** (upper-right panel), which is used to display all the services that have been highlighted from user-defined filters selected from the **Service Type and Filter Panel** (upper-left). The fourth component is the **Job Status Panel** (the lower-left panel), which is used to display the currently running jobs' status. The last major main component is the **Lightweight Web Browser** (the lower-right panel), which is used to display the original service developer's web page.

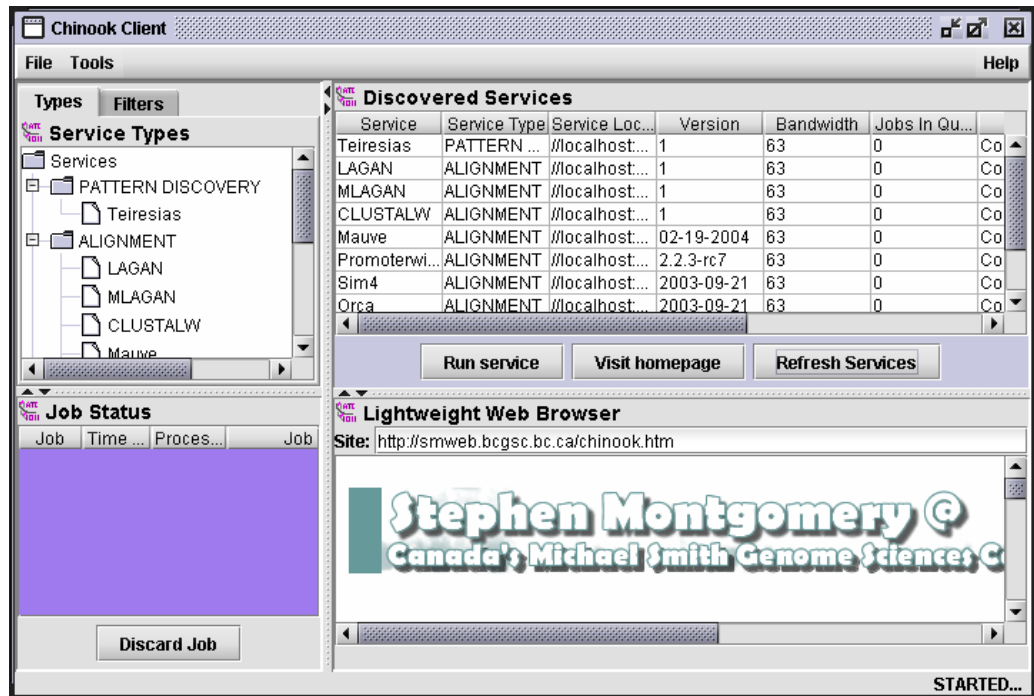


FIGURE 5.1 Chinook client starts up.

Step 2: Specifying a filter.

You can select the services you want to run by clicking the service name on the **Service Type and filter panel** pane. For example, you can click ALIGNMENT on the left panel; all the servers providing ALIGNMENT services will appear on the **Discovered Services Panel**. If there are too many servers providing the service, you can further filter the servers by providing some criterion in filter panel. To do so, click the **Filters** tab in the **Service type and filter panel**. Specify the filters you want by typing in the text field, then checking the checkbox of the corresponding text field. The services which fulfill the filter criterion will be displayed in the **Discovered Services Panel**.

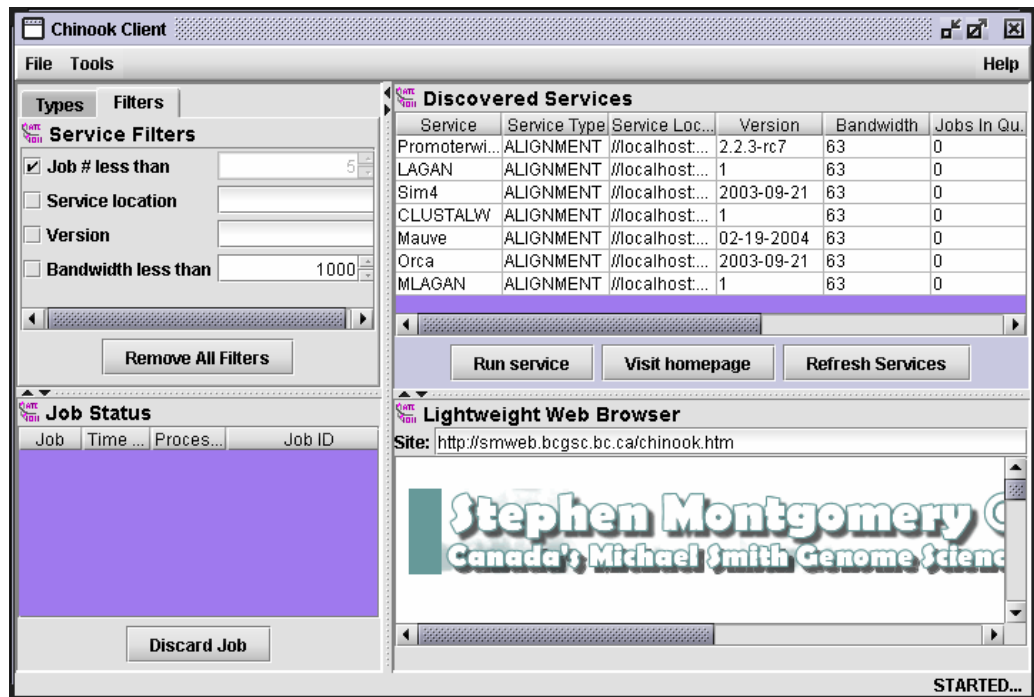


FIGURE 5.2 Specify filters to facilitate finding a service.

Step 3: Running a service on a server.

To run a service, first select the service you want to run in the **Discovered Services** panel by clicking the name of the service; then click the **Run service** button. Or you can right click on the service you want to run; then select **Run job on server** in the Popup menu. For example, if you select MLAGAN in the Discovered Services panel and then click **Run service** button, a window like the following will appear on the screen:

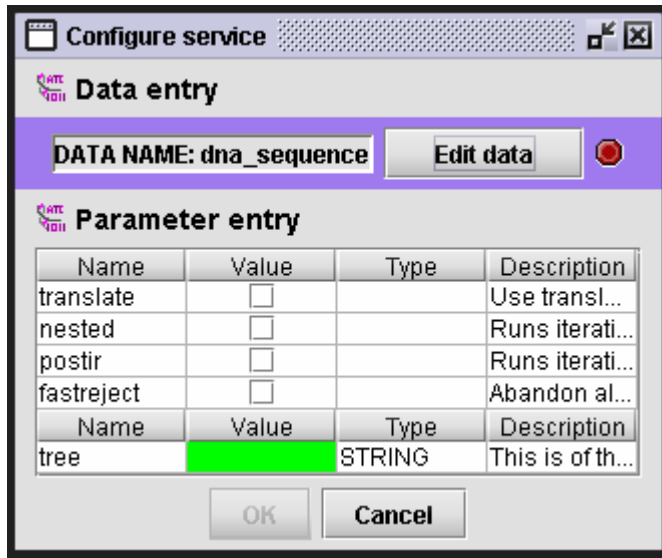


FIGURE 5.3 Configuring Service window. It is used to edit data and supply parameters.

Step 4: Editing data for your service

In the above window, the red dot to the right of the **Edit data** button indicates that the currently supplied data is invalid. You can specify the data you want by clicking the **Edit data** button. A window like the following will appear:

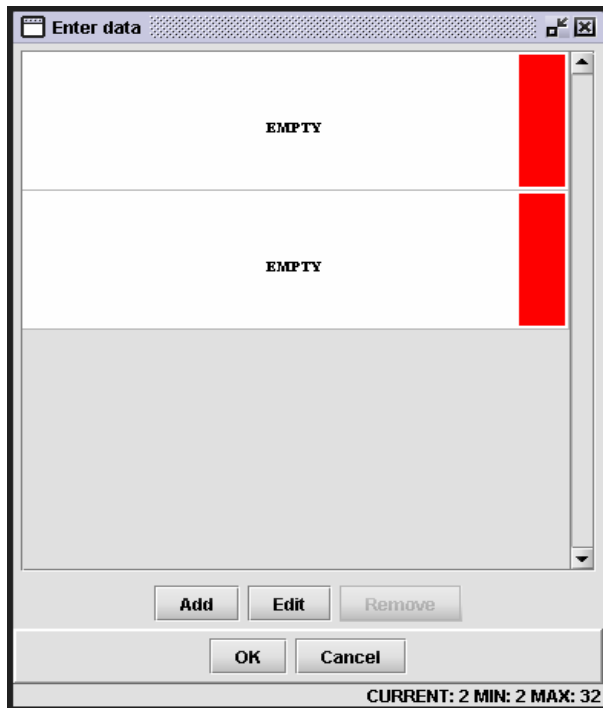


FIGURE 5.4 Enter data window. It is used to enter all the data the service needed.

In the above window, the red square on the right of the **data box** indicates that currently the data supplied is invalid. You can edit the data by select one of the **data boxes** by clicking on it and then clicking the **Edit** button. A window like the following will appear:

The screenshot shows a window titled "Enter data" with a "Select retrieval mode" sidebar on the left containing "DNA_LOCATION". The main area is purple and contains the following fields:

- Name: (red dot)
- Chromosome: (red dot)
- Start: (red dot)
- End: (red dot)
- Strand: 1 (dropdown, green dot)
- DatabaseType: ENSEMBL (dropdown, green dot)
- Species: Rattus_norvegicus (dropdown, green dot)
- Database: rattus_norvegicus_core_21_3b (dropdown, green dot)

A red-bordered box at the bottom contains the text "DATA INVALID". "OK" and "Cancel" buttons are at the bottom.

FIGURE 5.5 Enter data window. It is used to enter specialized data used by the service.

In the “**Enter Data**” window above, the red dot to the right of each text field and the red square under the window indicate that the data is invalid. You can point the mouse to the red square; a tool tip will tell you which part the data is invalid. After you have specified data, if the data entered is valid, the red square will change to a green square. This indicates the data now is valid. You can click the **OK** button to return to the previous window to edit another data box if there are some. After you finish editing all the data boxes, you can click the **OK** button to return to the **Configure Service** window. Now the red dot to the right of the **Edit data** button should become green to indicate the underlying data is valid. You are now ready to submit the job to the server.

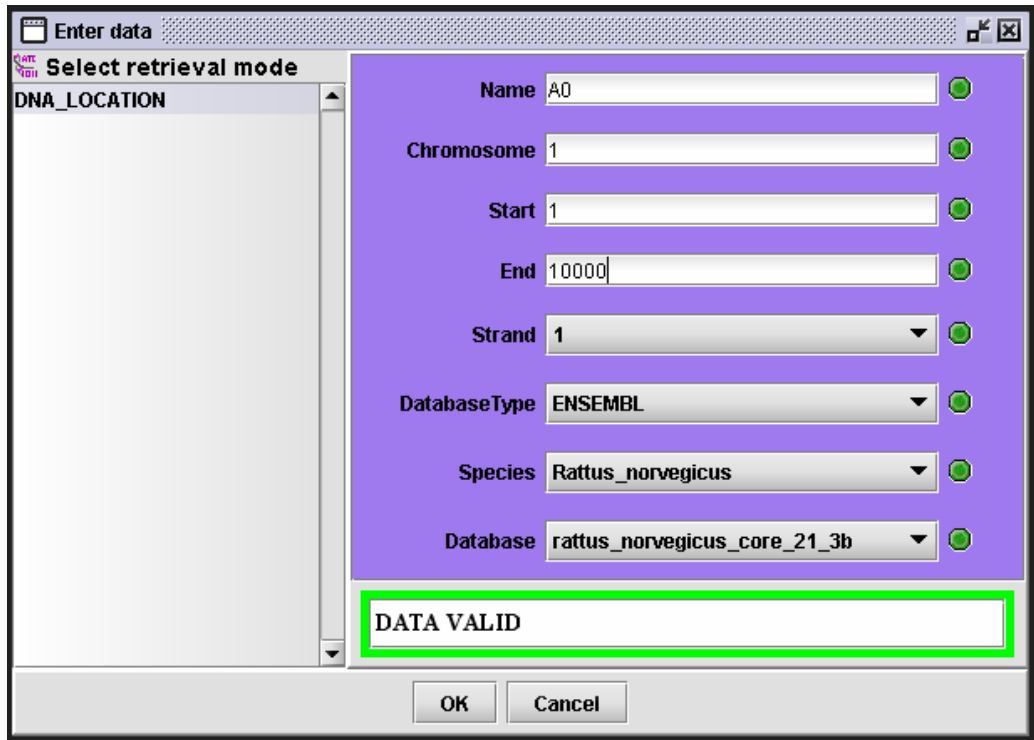


FIGURE 5.6 Enter data window. Enter the valid data.

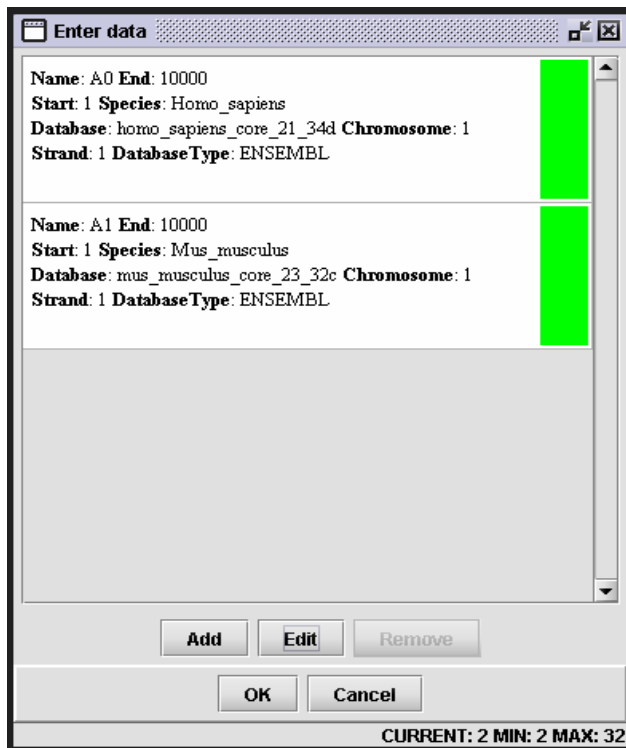


FIGURE 5.7 Enter data window. Add all valid data.

You can specify the parameters needed by the service by modifying the **Parameter entry** panel. Click the **OK** button on the **Configure service** window to run the service on server.

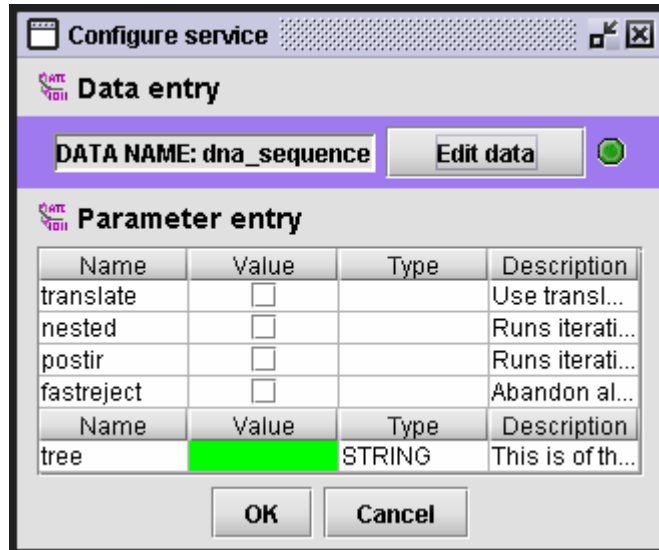


FIGURE 5.8 Configure service window. Specify parameters, and submit the job

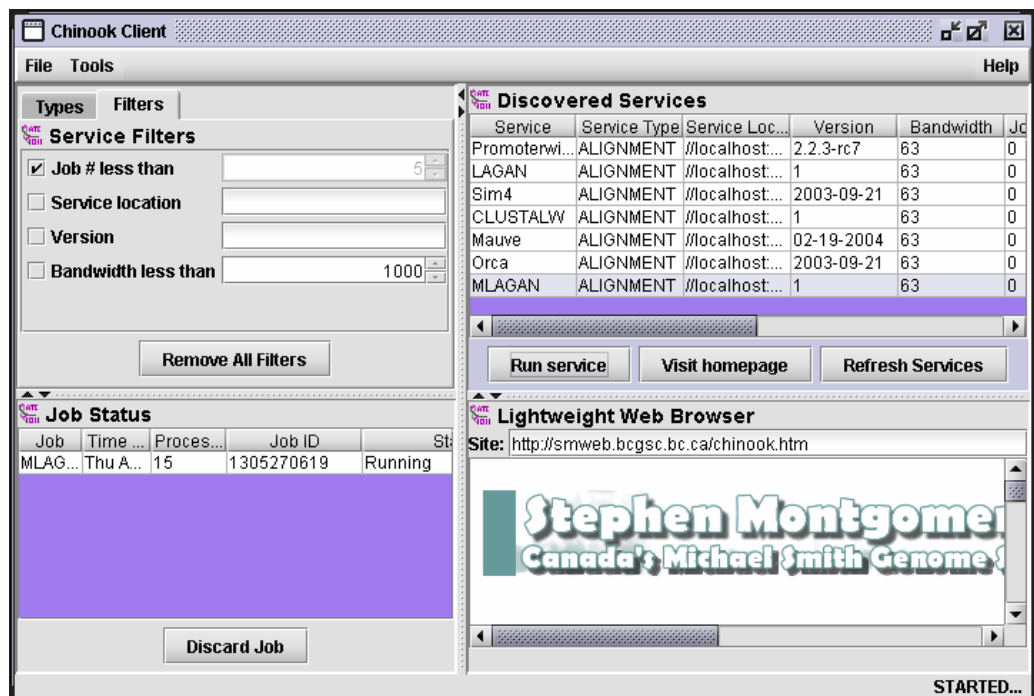


FIGURE 5.9 Running job. The job is running on the server.

You can run multiple jobs simultaneously by following the same procedure above. If for some reason you want to cancel a jobs you have submitted, just select the job you

want to cancel by clicking it in the **Job Status Panel**, and then click the **Discard Job** button. The job will stop running on the server, and will be removed from the table. You can do the same thing by right clicking on the job you selected, and then clicking on the **Discard job** item in the popup menu.

Step 5: Viewing the results

You can at any time view the job status by right-clicking on the job you selected, and then click the **View result files** on the popup menu. A window like the following will appear.

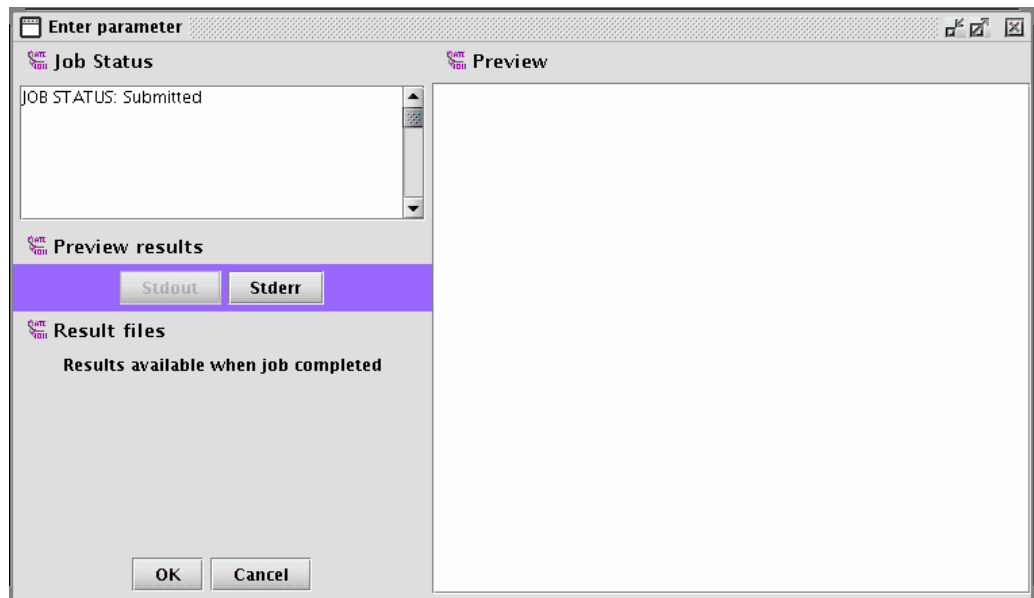


FIGURE 5.10 running job window.

After the job is completed on server, the **Job status** and **Result files** panel will be updated as below.

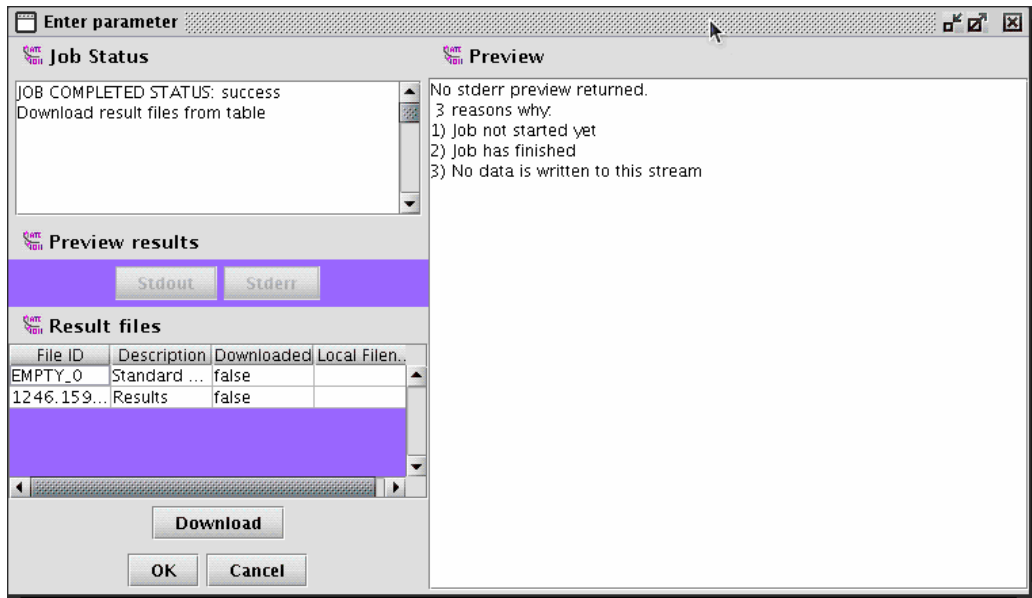


FIGURE 5.11 Job finished window. The job is finished on the server

In the above example, one of the table entries in the **Result files** is the standard error stream (NOTE: You can view the streams of processing jobs as they are executing on the server to check if normal processing is occurring). If an error occurs when the server is running the job, you can download the message to see what has caused the error. If the job is completed successfully, you can download the resultant files by selecting the appropriate table entries, and then clicking the download button.

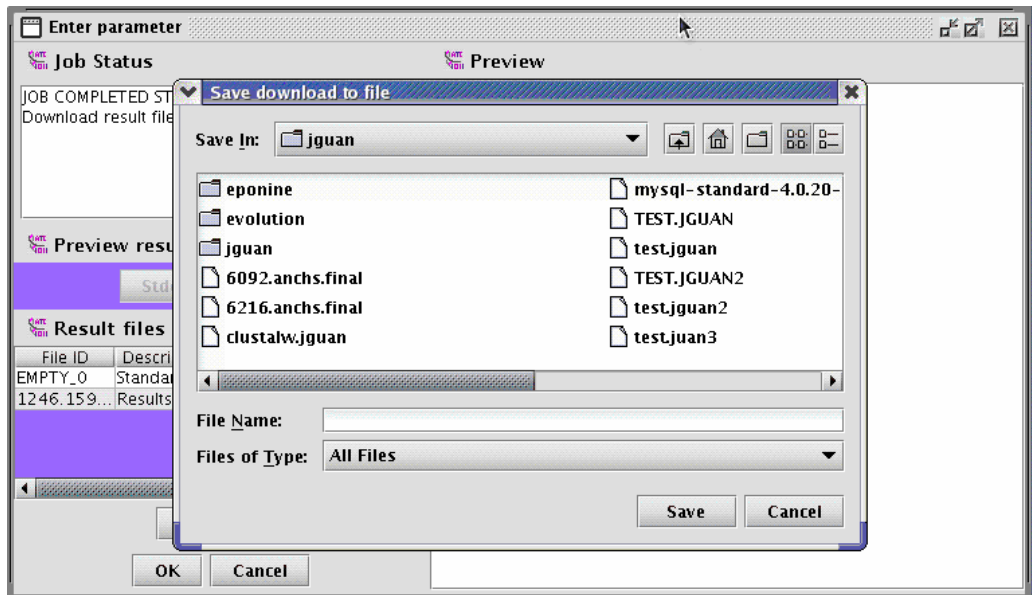


FIGURE 5.12 Downloading report window. The report is downloaded from server to local hard drive.

After downloading the file, you can use your favorite text editor to open the report and read it.

Step 5: Exiting Chinook

You can go to **File** menu and click **Exit** to exit Chinook Client. Or you can click the **close** button located on the right of the title bar of Chinook Client.

[Back to Table of Contents](#)

5.0.2 Adding a new Service

Currently, there are several dozen analysis services that have been integrated into Chinook. These range from DNA sequence alignment to gene regulation prediction algorithms. For a complete and updated list of services Chinook currently supported, Visit the Chinook website at <http://www.bcgsc.ca/gc/bomge/chinook/algorithms>.

The following example gives you an idea how to add a new service to Chinook Server. Let's look at LAGAN as an example.

```
<application>
  <name>LAGAN</name>
  <type>ALIGNMENT</type>
  <path>/opt/mlagan</path>
  <executable>lagan.pl</executable>
  <format>exe_path/executable dna_sequence parameter</format>
  <allow_stderr_preview>true</allow_stderr_preview>
  <allow_stdout_preview>true</allow_stdout_preview>
  <results_written_to_stdout>true</results_written_to_stdout>
  <parsing_class>
    ca.bcgsc.chinook.server.runner.alignment.Lagan
  </parsing_class>
  <output_path>/tmp</output_path>
  <description>
    Lagan is developed at Stanford by Mike Brudno
  </description>
  <creator>http://lagan.stanford.edu</creator>
  <version>1</version>
  <data_entry_set>
    <name>dna_sequence</name>
    <maximum_count>2</maximum_count>
    <minimum_count>2</minimum_count>
    <data_entry_type_name>DNA_LOCATION</data_entry_type_name>
    <data_entry_type_name>DNA_FILE</data_entry_type_name>
    <set_output_class_name>
      ca.bcgsc.chinook.parsing.setoutput.impl.DataEntrySetOutputter
      Impl
    </set_output_class_name>
  </data_entry_set>

  <parameter>
    <descriptor>chaos_STRING</descriptor>
```

```

    <regex_format>["](.+)["]</regex_format>
    <description>The contents of this string will be passed
as arguments to chaos</description>
    <user_defined>>true</user_defined>
  </parameter>
  <parameter>
    <descriptor>order_STRING</descriptor>
    <regex_format>"-gs ([0-9]+) -gc ([0-9]+) -mt ([0-9]+) -ms
([0-9]+)"</regex_format>
    <description>The contents of this string will be passed
as arguments to order
    </description>
    <user_defined>>true</user_defined>
  </parameter>
  <parameter>
    <descriptor>recurfl_STRING</descriptor>
    <regex_format>"(\([0-9]+\,[0-9]+\,[0-9]+\,[09]+\),)+</regex_format>
    <description>Used in recursive anchoring</description>
    <user_defined>>true</user_defined>
  </parameter>
  <parameter>
    <descriptor>translate_BOOLEAN</descriptor>
    <description>Use translated anchoring</description>
    <user_defined>>true</user_defined>
  </parameter>
  <parameter>
    <descriptor>bin_BOOLEAN</descriptor>
    <description>Output in binary format</description>
    <user_defined>>false</user_defined>
    <on>>false</on>
  </parameter>
  <parameter>
    <descriptor>mfa_BOOLEAN</descriptor>
    <description>Output in multifasta format</description>
    <user_defined>>false</user_defined>
    <on>>true</on>
  </parameter>
  <parameter>
    <descriptor>rc_BOOLEAN</descriptor>
    <description>
      Reverse complement the second sequence before alignment
    </description>
    <user_defined>>true</user_defined>
  </parameter>
  <parameter>
    <descriptor>fastreject_BOOLEAN</descriptor>
    <description>Abandon alignment if homology looks weak
    </description>
    <user_defined>>true</user_defined>
  </parameter>
</application>

```

All new application specs are defined between **<application>** and **</application>** tags. The **<name>** tag defines the application that is being run. This can be anything.

The **<type>** tag is more important. This is an ontological definition that marks the type of services class you belong to. It is planned that the website will carry a dictionary of the terms that wildly used. For now, the well-defined term is ALIGNMENT, VARIATION, MOTIF DISCOVERY, PATTERN DISCOVERY. The **<path>** tag simple points to the directory that the main service is in and **<executable>** tag holds the name of the application that will be run.

In the **<format>** tag, several terms are special and are replaced by appropriate values when the script is run.

- 1) exe_path is replaced by the contents of the **<path>** tag.
- 2) executable is replaced by the contents of the **<executable>** tag.
- 3) dna_sequence is replaced by the location of the sequence files.
- 4) parameter is replaced by the services specific parameters.

<allow_stderr_preview> tag defines if the standard error preview is allowed.
<allow_stdout_preview> tag defines if the standard out preview is allowed.
<set_output_class_name> tag defines the class name used to setup the output.
<parsing_class> tag defines what class will parse the output file. Currently supported parsing classes are: **ca.bcgsc.chinook.server.runner.alignment.Lagan** for Fasta, **ca.bcgsc.chinook.server.runner.alignment.Clustalw** for GCG. If a parsing class isn't defined, you will have to make one in the runner package of the Chinook server code.

The **<output_path>** tag points to your temporary directory for formatting files. The **<description>** tag points to a description of the service. The **<creator>** tag is the website of the original author of the services implementation (not the service providers). So in the case of Lagan, it points to the site at Stanford. The **<version>** is the version of the application.

The **<data_entry_set>** tag defines the data formats. The first tag **<name>** defines the name of the data, which is used in **<format>** tag. The **<maximum_count>** defines the maximum number of sequences. If there are no maximum number of sequences can be supplied to the application. This tag does not need to be defined. The **<minimum_count>** defines the minimum number of the sequences being supplied to the application. **<data_entry_type_name>** defines the name to get the sequences. **<set_output_class_name>** defines what class is used to setup the output.

The **<parameter>** tag is another special tag in the XML description of your service. These tags define what parameters you want your client to input, what parameters you'd prefer they didn't, and what default values you'd like to maintain. The **<parameter>** tag offers extensive control over how your client uses your service.

The first part of the **<parameter>** tag if the **<descriptor>** tag. This tag defines what type of parameter it is. Fundamentally, there are four types STRING, BOOLEAN, OPTION, and FILE. When defining the **<descriptor>** tag, define it as the name

than the type, i.e. `tree_STRING` tells Chinook that you have a parameter called `tree` that needs a string whereas `translate_BOOLEAN` tells Chinook that you have a parameter that is either there or it is not; for instance,

```
mlagan -tree "mytree" -translate
```

The `<regex_format>` tag describes the regular expression that you want your input string data to match. This is a security feature that allows you to guarantee that parameters will be inputted in the way that you expect to get them – as users are prevented from entering parameters that don't match. This tag is not required however.

The `<description>` tag describes to the user what this parameter does. The `<user_defined>` tag tells Chinook whether you want the user to be able to change this parameter; it has two options true or false. The `<use_equals>` tag tells Chinook whether the parameter is of the form `-tree=value` or `-tree value`. It takes two options true or false. The `<on>` tag tells Chinook whether this boolean parameter is active or not by default. Finally, the `<default_value>` tag holds the data that you want this parameter to input, i.e the default string data. It is very possible as a service provider to use these tags in way that doesn't make sense. Be very careful about what you want users to do and what the default parameters are. If you find that something is missing to fully describe your input parameters, e-mail me at smontgom@bcgsc.bc.ca

After, you have defined your new service you should be able to share it with the world using Chinook. Visit our online service at <http://smweb.bcgsc.bc.ca/cgi-bin/chinook-status.pl> to see if our discovery service has picked it up.

[Back to Table of Contents](#)

5.0.3 Setting up a Chinook server node

This walkthrough will guide you through setting up a Chinook server.

To make Chinook available to Internet users, you will have to have as a prerequisite:

- 1) A computer where bioinformatics applications can be installed and run (i.e. you will not be able to provide ClustalW analyses if your computer cannot run it)
- 2) Open required ports on the computer. You will need ports 9700 and 9701 open for JXTA communication. You will also need either port 1099 (for RMI mode) or port 8080 (for Web Services mode). These are the default ports; other ports can be selected in place of these in case of overlapping services (you will need to edit the advertisement and resource files if you are not using a default port).
- 3) A working directory. Ideally, your computer will have at least 100MB of hard-drive space to store temporary files.

To set-up a Chinook server node:

- 1) Ensure you have the right hardware dependencies (see above).
- 2) Install Java.
 - a. Go to <http://java.sun.com>
 - b. Look for Downloads
 - c. Download the 1.4.x version of the J2SE JDK.
 - d. Set the JAVA_HOME environment variable to the installation directory of Java (i.e. export JAVA_HOME=/usr/lib/java). You may want to do this in a configuration script so that this environment variable is preserved.
- 3) Install Chinook (see Section 2).
- 4) If you are planning to use the Web Services version of Chinook, you will need to install Tomcat and Apache Axis. Read the inset for instructions on how to do this.

Installing Tomcat and Apache Axis:

APACHE TOMCAT

- 1) Go to <http://jakarta.apache.org/tomcat/>.
 - a) Look for Downloads → Binaries.
 - b) Get the latest stable build of Tomcat.
 - c) We used 5.0.28 for our latest installation
- 2) Unpack Tomcat
- 3) Set the **CATALINA_HOME** environment variable
 - a) This point to the installation directory of Tomcat (i.e export **CATALINA_HOME=/home/chinook/tomcat/jakarta-tomcat-5.0.28**)

APACHE AXIS

- 4) Go to <http://ws.apache.org/axis/>.
 - a) Look for Downloads → Releases.
 - b) Get the latest stable build of Axis
 - c) We used 1.1 for our latest installation
- 5) Once Axis is installed, copy the contents of the webapps/axis directory to the webapps directory of your Tomcat installation. Restart Tomcat (issue a shutdown.sh and startup.sh from the bin/ directory of your Tomcat installation).
- 6) You are now able to verify that Axis is installed. Go to the webpage of your service (i.e. <http://mymachine:8080/axis>, where mymachine is your machine name). Click on Validate. If any of the core jars are missing, they will have to be downloaded and installed into **WEB-INF/lib/** directory of **webapps/axis** (in its Tomcat directory).

JAF

- 7) If you are missing the activation.jar, download it from <http://java.sun.com/products/javabeans/glasgow/jaf.html>
- 8) Copy the jar to the **webapps/axis/WEB-INF/lib/** folder in your Tomcat directory.

RUNNING USING RMI

For RMI, no dependency installations are required except a valid Java installation.

If you are developing/modifying Chinook and running it from an IDE or directly from the source (not from Ant or an Installer) you will need to explicitly set the RMI security manager, and specifying the codebase. This can be done by adding to your VM parameters the following lines changing them where appropriate to match your chinook directory (note that the codebase parameters are separated by a space):

```
-Djava.rmi.server.codebase=file:///home/smontgom/jbproject/chinook/classes/  
file:///home/smontgom/jbproject/chinook/lib/filewire.jar  
  
-Djava.security.policy=/home/smontgom/jbproject/Chinook/resources/chinookRMI.policy
```

- 5) For the Web Services version of Chinook only, you must now deploy the Chinook code. This is done by running the Ant task `deploy` from the Chinook installation directory. If the **CATALINA_HOME** environment variable has not been set, this task will fail.
 - a. Run the command: `ant deploy`
 - b. If Ant is not installed, follow the installation instructions below.

Installing ANT

APACHE ANT

- 1) Go to <http://ant.apache.org/>
 - a. Find Download → Binary Distributions
 - b. We used 1.6.2 for our latest installation
- 2) Add Ant to your PATH environment variable
 - a. Export `PATH=$PATH:/home/chinook/ant/apache-ant-1.6.2/bin/`
 - b. Replace the ant path above for your own installation directory.
 - c. You should now be able to call Ant from any directory

- 6) Before you run the server, in either RMI or Web Services mode, you will need to configure the server for your machine. Go to the Chinook installation directory.
 - a. Editing the files in the **resources/** directory
 - i. The most important file to edit is **applications.xml**

- ii. You will want to comment out the protocol block you are not using and comment out the protocol block you are using. For instance, in RMI mode the following would appear:

The red code is commented out, the green code is uncommented. This server is operating in RMI mode in this configuration.

```
<!--WEBSERVICES
<protocol>WSDL</protocol>
<uri>http://localhost:8080/axis/services/ApplicationServerImpl</uri>

<using_filewire>true</using_filewire>
<filewire_uri>http://localhost:8080/axis/services/FileWireImpl</filewire_uri
>
-->

<!-- RMI -->
<protocol>RMI</protocol>
<uri>//localhost:1099/ApplicationServerImpl</uri>

<using_filewire>true</using_filewire>
<filewire_uri>//localhost:1099/FileWireImpl</filewire_uri>
<!-- -->
```

- iii. **IMPORTANT:** Change the name of the `uri` to your machine name. Do not use localhost.
 - iv. Change the publisher information to reflect your user information. This will allow users of your server to contact you. This information should also be set in more detail in the `server-info.xml` file in the same directory.
 - v. **IMPORTANT:** We have not installed any new services into Chinook. When new services are added, they are described in the **applications.xml** file. If this file contains services, you should comment them out as your server would end up advertising services that do not exist at your location.
- b. Editing the files in the **advertisements/** directory
 - i. The advertisements that Chinook uses are specified in **advertisement-config.xml** in the **resources/** directory.
 - ii. To ensure that you are advertising the right endpoint for your services, edit the advertisement implementation files. Change it from localhost to your machine name.
- 7) That is all there is to it. You **MUST** still install services. But to run Chinook start `ant p2p-start` then wait a few seconds (until the `p2pNode` has

found a rendezvous ~ 10-15 seconds) then type `ant server-start`.
NOTE: Check the Axis configuration page to ensure that your service was deployed. If it was not, first try stopping and restarting the Tomcat server.

- 8) Test out your services by running a client.

Troubleshooting:

If you are not able to see the deployed services, look in the Tomcat log/ directory to determine the source of the error.

The server-config.wsdd file is not found.

Copy this file from elsewhere in your Tomcat installation (it will likely be in the work/ directory).

I get a 401 error from ANT and the log says: - Rejected remote access from host /0:0:0:0:0:0:1

The server-config.wsdd file needs to allow remote administration. Set `<parameter name="enableRemoteAdmin" value="true" />` for the AdminService.

Other error

Send the tailing lines of your log files and the ant execution information to chinook@bcgsc.bc.ca.

[Back to Table of Contents](#)

5.0.4 Running a batch Perl job

The Perl interface to Chinook allows you to add analysis capabilities directly into your scripts. This walkthrough will outline how to find services, submit jobs, and read reports from using the Perl interface.

Follow steps:

- 1) Configure your Perl environment. (see [3.0.4.1](#))
- 2) Start the Chinook Client in batching mode (see [3.0.4.2](#)).
- 3) The example Perl scripts for Chinook batching are located in the `perl/t/` directory from your Chinook installation directory.
- 4) **How to Find Chinook Services using Perl.** One of the first programmatic tasks is to discover what services are currently available for

running over Chinook. There are two ways to input discovered services into your Perl script. This can be either performed by parsing the batch directory using the **Bio::Tools::Run::Chinook::BatchDirectory** module or by asking the Chinook Client directly using the **Bio::Tools::Run::Chinook::ChinookManager** module.

a. To parse the batch directory:

```
my $batch_directory =
    Bio::Tools::Run::Chinook::BatchDirectory->new(
        batch_directory => "/home/smontgom/batch/batch");
my $services_ref = $batch_directory->getAllServices();
```

b. To use the **Bio::Tools::Run::Chinook::ChinookManager**:

```
my $chinook_man =
    Bio::Tools::Run::Chinook::ChinookManager->new(
        machine_name => "localhost", port => "7999");
my $services_ref = $chinook_man->getServices();
```

Both of these methods get the current services that are available. (Each returns a list of **Bio::Tools::Run::Chinook::Service** objects) However, only the **Bio::Tools::Run::Chinook::ChinookManager** is guaranteed of being current as service descriptions are not removed from the batch directory unless manually removed. The recommended method is to use the **Bio::Tools::Run::Chinook::ChinookManager** to get current information whenever possible.

- 5) Once a service has been selected, you will need to access a batch file (create a **Bio::Tools::Run::Chinook::Batch** object) for that service to get the required parameters and supported databases.
- 6) **How to Get Information about Required Data and Parameters using Perl.** To create a **Bio::Tools::Run::Chinook::Batch** object, call the following (the filename of the batch file can be accessed from the **Bio::Tools::Run::Chinook::Service** object or it will have to be discovered in the batch directory if the **Bio::Tools::Run::Chinook::ChinookManager** doesn't provide the information)

```
my $batch = Bio::Tools::Run::Chinook::Batch->new(
    batch_filename => $batch_filename);
```

Once, you have a `Bio::Tools::Run::Chinook::Batch` object, you can interrogate the required DEOSets (Data Entry Object Sets) that are required to run the Service.

NOTE: Data Entry Object Sets (DEOSets). DEOSets are the data requirements for various services. A DEOSet contains information about what types of data objects a server is expecting, how many of them it requires, and the data itself (in the form of a DEO, Data Entry Object). Each DEOSet has a name that references it to the server as the data that is contained in a DEOSet is specifically manipulated to allow the associated service to access it in a desirable way. To use a DEOSet, you must first determine what DEO types are acceptable. Examples of the types are “DNA_LOCATION” or “DNA_FILE”, each describes a specific type of data that the server knows how to interpret. In this case, the server can either take a file containing DNA sequence or the specific coordinates of a genomic sequence (from one of the supported databases on the server). You will need to ask the server to provide an outline of the data that is required. This is done in Perl by using the `Bio::Tools::Run::Chinook::ChinookManager` to get the associated DEO for a DEO name, as so:

```
my $deo = $chinook_manager->getDEO("DNA_LOCATION");
```

- 7) Fill in the required DEOSets. The process to do this is in the test scripts in the perl/t/ directory. Essentially, first get the DEOSet objects from the `Bio::Tools::Run::Chinook::Batch` object. Iterate through each DEOSet and determine how many DEOs are required and what the allowed DEO types are (i.e. DNA_LOCATION, etc). Once you have found a DEO type that is suitable for your purposes, get the DEO object from the ChinookManager (see the Note above). From the DEO object, you can access all the attributes of this DEO and fill them in. (These will be validated when you submit the job if a mistake is made). You can set various properties like the DEO order and name to specifically reference how and in what order the server should process the data. (A negative number for the order means that the client doesn't care what order the data is in). See the testBatch.pl scripts for examples of how this is done!
- 8) Once the data has been filled in and set, the required parameters need to be set.
- 9) **How to Set Required Parameters using Perl.** The parameters that are required for any given service are accessible from the `Bio::Tools::Run::Chinook::Batch` object. An example of how to set all the Boolean parameters to false is described below.

```
my $parameters_ref = $batch->getParameters();
```

```

my @parameters = @$parameters_ref;
foreach my $parameter (@parameters) {
    if ($parameter->getType eq "BOOLEAN") {
        $parameter->setValue("false");
    }
}

```

- 10) **How to Run a Chinook Job using Perl.** To run a job in Chinook using the Perl interface, you finally need to create a `Bio::Tools::Run::Chinook::QueueBatch` object. Once the parameters and DEOSets have been set, this can be done as below:

```

my $queue_batch =
Bio::Tools::Run::Chinook::QueueBatch->new(
    deosets => $deosets,
    parameters => \@parameters,
    batch => $batch,
    filename =>
        "/home/smontgom/batch/batch_queue/chinook.LAGAN");

```

The `Bio::Tools::Run::Chinook::QueueBatch` object requires a filename which will become the prefix for the XML file containing all the information required to run the job over Chinook. This file is written by calling:

```

my $id = $queue_batch->writeQueueBatch();

```

Then the `ChinookManager` is used to point the Chinook Client at the batch queue file for processing on the server. An example of this is below:

```

$chinook_man->processQueueBatch($queue_batch);

```

- 11) **Accessing Reports using Perl.** The ID that was returned when the `Bio::Tools::Run::Chinook::QueueBatch` object was written to file is used to reference that file to resulting report files that are written to the batch reporting directory. To monitor completion of reports, you can periodically poll the reporting directory (substituting your reporting directory in place of the one provided below). An example of this is below.

```

$filename = $chinook_man->isReportReady($id,
    "/home/smontgom/batch/batch_reporting/");

```

- 12) If the filename is defined, the report has been written to the file.

- 13) **Getting the Bio::Tools::Run::Chinook::Report object.** To get the Report object, once the filename has been defined, call:

```
my $report =  
    Bio::Tools::Run::Chinook::Report->new(  
        report_filename => $filename);
```

- 14) From the report object, you can get information about warnings, errors, or the information required to download the result files from the server.

- 15) **Downloading Results from the Report.** To download the results from the report file, first determine the canonical output file. This is the file that the results are written to (you can also access other files that describe how the job was run and what was available on various streams). To download results to a sample file, follow the example below.

```
my $report_file = $report->getCanonicalReportFile();  
  
my $sample_file = "/tmp/results.out.steve.3";  
  
$chinook_manager->downloadFile(  
    $report_file->getFileId(),  
    $report->getServiceLocation(),  
    $sample_file);
```

- 16) **Congratulations.** That should cover the basic process for running jobs via Chinook. There are still lots of steps required that we hope to reduce the complexity of. There is also lots of uncovered functionality that can be observed by looking at the modules and reading the associated Perldocs. But, with a little bit of effort, your scripts and users can access state-of-the-art algorithms without having them downloaded.

[Back to Table of Contents](#)

6.0 Further Information

Chinook is funded by Genome Canada as part of the Bioinformatics of Mammalian Gene Regulation grant. Stephen Montgomery is a Ph.D. graduate student in Genetics at the University of British Columbia. He is funded by the Michael Smith Foundation for Health Research. His work is performed primarily at Canada's Michael Smith Genome Sciences Centre as part of the Gene Regulation Informatics team. Steven Jones is the Head of Bioinformatics for the CMSGSC. He is a Scholar of the Michael Smith Foundation for Health Research. Currently, there is no source code available for Chinook. The source though is licensed under Creative Common's Attribution-Non-Commercial license and is freely available on request to chinook@bcgsc.bc.ca or can be downloaded using any of the options from Section 2 of this document. If you have any questions and find any bugs in Chinook, please email us.

6.0.1 Mailing List

The Chinook mailing list is a low-volume regulated list that broadcasts weekly development announcements. We recommend you to sign up the mailing list at <http://www.bcgsc.ca/mailman/listinfo/chinook>. You will get the latest information about Chinook (including upgrade, bug fix). You can also view the Archive at <http://www.bcgsc.ca/pipermail/chinook/>.

[Back to Table of Contents](#)

6.0.2 Authors

Montgomery SB, Fu T, Guan J, Lin K, Jones SJM (in preparation).

Chinook Internal:

Chinook Service Developers:

Monica Sleumer, Keven Lin, Tamara Astakhova, Jun Guan, Maik Hassel, James Kennedy, Eddy Tsang, Yvonne Li, Tony Fu

Other thanks:

Asim Siddiqui, Misha Bilenky, Gordon Robertson

Chinook External:

Jonathan Lim, Wyeth Wasserman, David He, Sohrab Shah, Francis Ouellette

[Back to Table of Contents](#)

6.0.3 Known Problems

1. Web pages may not be displayed properly in the **Lightweight Web Browser**. This is because Chinook is a Java application, and JEditorPane, which is used to display web pages, only supports Html 3.2 currently. Any web pages created using html version above 3.2 will likely not be displayed properly.

[Back to Table of Contents](#)

6.0.4 License



Chinook is licensed under a [Creative Commons License](#).

[Back to Table of Contents](#)