

MacroModel™ 7.1

MacroModel User Manual Appendices 1 and 3 Updated for Version 7.1



1500 SW First Ave. Suite 1180
Portland, OR 97201
Phone: (503) 299-1150
Fax: (503) 299-4532
help@schrodinger.com

© 2000 Schrödinger, Inc.
All rights reserved.
January 2001

Derived from material which is
© Columbia University

Table of Contents

Appendix 1: Installing MacroModel	5
MacroModel Directory Structure	5
Environment Variables	5
INCLUDE directory paths	6
SGI and IBM Installation	6
Appendix 3: Setting Up BatchMin Communication	7
Summary	7
How To Install Entries In hostfiles.dat	8
Adding Entries For Locally Executing Jobs	8
Troubleshooting	9
Symptoms	9
Actions	10
NFS-Style Remote BatchMin Installation Instructions	10
How the NFS-Style Remote BatchMin Facility Works	10
How to install entries in hostfiles.dat	10
Establishing a shared directory with NFS	11
Notes	15
BatchMin Network Server Installation Instructions	16
Overview	16
Installation	17
Conventions	17
Step1: Informing inetd	18
Step 2: Get rsh running	21
Step 3: Try Out rbm and 'jump start' the daemon	22
Step 4 (Optional): MacroModel—Accessing the Server From the Graphical User Interface	24
Some Final Notes	24
Troubleshooting	25
Using the RPC Network Connection to Run BatchMin Manually	27
Installing Additional BatchMin Network Server Services	29
BatchMin Force Field, Solvent and atom.typ Files	30

Table of Contents

Configuring bminrd, the BatchMin Network Server, to Start Automatically at Boot Time on an SGI	31
---	----

Appendix 1: Installing MacroModel

1.1 MacroModel Directory Structure

As of release 7.1, the top-most directory where MacroModel™ is loaded is referred to as \$SCHRODINGER. The \$SCHRODINGER directory contains multiple subdirectories corresponding to any installed Schrödinger software products. The version of MacroModel that you have installed will be located in one of the subdirectories.

Environment Variables

The environment variable \$SCHRODINGER needs to be set to the directory into which MacroModel was loaded. If \$SCHRODINGER is not set, the macro-model shell script, when executed, will set it to the current working directory by default. Setting \$SCHRODINGER to the MacroModel installation directory will allow you to launch MacroModel from any directory.

As of release 7.1, the environment variables \$MMSITEDIR and \$MMEXECDIR should no longer be used. These variables are now set automatically by the MacroModel and Maestro GUIs upon startup. Check to make sure that the variables are not being set by your login shell script. Only the SCHRODINGER variable need be set.

INClude directory paths

There is one include directory, INC1. This will default to \$MMSITEDIR. The default value can be superseded by setting INC1 paths in \$SCHRODINGER/macromodel-<X>/data/mmodsite.dat, where <X> is a version number (e.g. v71007). INC1 specifies the location of fragment template files, button help file (mmtplx.doc), and volume initialization files.

SGI and IBM Installation

The installation instructions are now in the *Schrödinger, Inc. Product Installation Guide*.

If energetic calculations using BatchMin are to be run on a remote computer, see on page 7 after having installed MacroModel.

Appendix 3: Setting Up BatchMin Communication

3.1 Summary

MacroModel - BatchMin communication is typically achieved through the file system. If BatchMin is run on the local computer (i.e. the same computer that MacroModel is running on), typically no changes to the default installation are required. However, in some cases the default installation must be modified to allow for, say, interaction with a queuing system or additional execution of commands before or after a job is run. In this case, additional entries must be added to the `hostfiles.dat` file. This file is located in a directory based on `$MMSHARE_EXEC`. Let us call this location *MMSHARE DATA DIR*. *MMSHARE DATA DIR* is automatically determined and set in the MacroModel GUI and the Maestro GUI startup scripts. In release 7.1 this location will be `$SCHRODINGER/mmshare-<X>/data`, where `<X>` is a version number (e.g. v10016).

If BatchMin is run on a remote computer (i.e. not the computer where MacroModel is running), then one of two methods may be used to set up a connection to the remote BatchMin. The first method (we will refer to this as NFS-style) requires that the remote machine NFS mount the local file system. The other method (called the server method) involves installing the BatchMin Network Server.

3.2 How To Install Entries In `hostfiles.dat`

Each entry in the `hostfiles.dat` file consists of two lines. The first line is the name of a script file or the name of a host. The second line (which **MUST** be included) contains either nothing, a directory name, or a '*' followed by an optional program name. Entries for jobs which are to be run locally should consist of the name of the script (which will be used to run the job) in the first line, and nothing in the second line. When creating an entry for NFS-style remote jobs (see below), the first line should contain the name of the script and the second line should contain the name of a directory. When creating an entry for remote server jobs, the first line should contain the name of the host where jobs are to be run and the second line should contain a '*', which may or may not have after it the name of the BatchMin executable to run. If there is no name, a default is used.

Every time a user clicks on the **Host** button on the MacroModel GUI, MacroModel first looks in the local directory for a `hostfiles.dat` file. If it finds one, it uses the information in this version of the file. Otherwise it looks in the *MMSHARE DATA DIR* directory for `hostfiles.dat`.

3.3 Adding Entries For Locally Executing Jobs

Entries for jobs that are to be run locally can be added as follows. First, create a script that can be used to start up BatchMin. An example is `bmin` in the `$$SCHRODINGER` directory. Note that MacroModel always passes the base name of the job to a script.

Once you have created a script file, test it to make sure that it can be used to run a BatchMin job from the prompt in a UNIX shell. If it is not working, check that:

- the script has sufficient execute permissions
- the script is handling the name it was passed correctly (the script should be run using a syntax like: `myscript <job>`)

To add your working script, edit the file `hostfiles.dat`, adding the name of the script on the first line of the new entry and an empty line as the second line. The placement of the new entry is important; putting the entry in the first position will make it the default.

From within the MacroModel GUI, clicking the **Host** button should produce a scrolling list that includes your new entry. Select your entry. You should not get any error messages or warnings. If you do, look for and fix any errors in the `hostfiles.dat` file. Possible errors include incorrectly spelled script names or scripts that reside in an incorrect location. If there are no errors, try to start up a job.

3.4 Troubleshooting

Symptoms

The following is displayed in the Message Window:

```
Submitting BatchMin job
Waiting For First Intermediate Structure
Waiting For First Intermediate Structure
Waiting For First Intermediate Structure
Waiting For First Intermediate Structure
Waiting For First Intermediate Structure
Waiting For First Intermediate Structure
Waiting For First Intermediate Structure
Waiting For First Intermediate Structure
Waiting For First Intermediate Structure
Waiting For First Intermediate Structure
Batchmin Job Not Confirmed
Checking .log File .....
.log File Open Failed.
```

This almost always means that the job failed to start. Possible reasons are:

- the script you created did not have execute permissions
- the script you created did not start up BatchMin properly (check path, name, etc.)

- the script you created started up BatchMin in such a way that BatchMin did not write its output files to the current directory
- the entry in the `hostfiles.dat` file is incorrect - perhaps *two* lines for the entry (with the second empty) were not added, or the script name was misspelled

Actions

If you did get a log file (i.e. you did NOT get the above '.log File Open Failed' message), then check for further information by clicking **ComFI** and choosing 'Display BatchMin .log Output File'. If there is no .log file or no further information in it, check and fix any of the above listed symptoms. On an SGI, if you get the message 'killed' in the log file, then you will need to enable virtual (logical) swap space. For instructions on how to do this, read `$SCHRODINGER/docs/swap.txt`.

3.5 NFS-Style Remote BatchMin Installation Instructions

How the NFS-Style Remote BatchMin Facility Works

How to install entries in `hostfiles.dat`

Create Start-up Shell Script Files

1. Ensure that rsh is installed and operating correctly. If MacroModel is running on computer *local_iris* and BatchMin is to run on computer *remote_iris*, then test rsh by entering the following command in a UNIX shell:

```
rsh remote_iris -l <login_id> ls
```

This command will print a listing of the contents of the home directory for user `<login_id>` on *remote_iris*. If this command fails, correct the problem before continuing.

2. Create a shell script of the following form:

```
#!/bin/csh -f
rsh remote_iris -l <login_id> <batchmin exe dir>/
bmin < $1.com >& $1.log
```

(<batchmin exe dir>/bmin refers to an executable BatchMin image compiled for and located on the remote machine)

Note: this shell script can be tailored to the local environment as necessary. An example stub can be copied from `$SCHRODINGER/macromodel-<X>/data/remote_bmin` where <X> is a version number (e.g. v71003).

When MacroModel executes the above shell script, the name of the command file will be passed as the first parameter. If the shell script were named `run_batchmin` and the BatchMin job named `peptid_1`, MacroModel would execute:

```
run_batchmin peptid_1
```

Use this command to start a BatchMin job and test the shell script.

3. Put the shell script in the local directory or in `$SCHRODINGER`.
4. Use the name of the shell script as the first line of an entry in `hostfiles.dat`.

Establishing a shared directory with NFS

The second line of each `hostfiles.dat` entry associates a mount point with each start-up shell script. In some cases this line is left blank. When starting BatchMin, MacroModel concatenates the `hostfiles.dat` mount point entry with the current working directory (i.e. the directory where MacroModel is executing) to create a directory path. This path is put in the first line of the BatchMin command file, and represents the remote computer's link to the local directory where MacroModel executes. The files that MacroModel and BatchMin use to communicate are created in this directory.

Appendix 3: Setting Up BatchMin Communication

There are two ways to mount the remote file system: (assume two computers—*local_iris* and *remote_iris*)

1. The *remote_iris* mounts the *local_iris* file system. Use this method if BatchMin files should be created on the computer that is executing MacroModel.
2. The *local_iris* mounts the *remote_iris* file system. Use this method if BatchMin files should be created on the computer that is executing BatchMin.

Case 1

The *remote_iris* computer has remotely mounted the *local_iris* computer file system. The file system must be exported with write privileges.

Suppose *local_iris* has a file system

```
/usr
```

and that *remote_iris* mounts this file system as

```
/local_iris
```

Create a soft link on *remote_iris* as follows:

```
mkdir /iris_link  
ln -s /local_iris /iris_link/usr
```

In `hostfiles.dat` the corresponding entry is:

```
remote_bmin_startup_script  
/iris_link
```

Suppose a user runs MacroModel on *local_iris* from the directory `/usr/people/chemist`.

Upon starting BatchMin, MacroModel will concatenate the mount point entry and the current working directory to create `/iris_link/usr/people/chemist`.

This path will be put into the BatchMin command file on the first line. It will be the case that:

```
/iris_link/usr/people/chemist
```

is the same as

```
/local_iris/people/chemist
```

on *remote_iris*; which is the same as

```
/usr/people/chemist
```

on *local_iris*. MacroModel and BatchMin will communicate through this directory.

Appendix 3: Setting Up BatchMin Communication

A single `hostfiles.dat` entry and start-up shell script can be used by any user in

```
/usr
```

on *local_iris*.

Case 2

The *local_iris* computer has remotely mounted the *remote_iris*' computer file system. The file system must be exported with write privileges.

Suppose there is a file system on *remote_iris*

```
/usr
```

and this file system is mounted remotely by *local_iris* as

```
/remote_iris
```

Create a soft link on *remote_iris*

```
ln -s /usr /remote_iris
```

Leave the `hostfiles.dat` mount point entry blank. MacroModel can now execute on *local_iris* from the remote file system:

`/remote_iris`

and BatchMin files will be created on the *remote_iris* file system.

Notes

- 1} In both cases, MacroModel executes on *local_iris*; in case 1 on a local file system, in case 2 on a remotely mounted file system. In both cases, BatchMin is started by executing a rsh command.
- 2} In both cases, a single `hostfiles.dat` entry and BatchMin account on the *remote_iris* can be shared by multiple users. The force field files can be given read only protections and access to the account can be controlled with the `.rhosts` mechanism.
- 3} Soft links can be avoided by using the following naming conventions when mounting remote file systems:

If the local file system is

`/usr`

name the remote file system

`/local_iris/usr`

(i.e.: *remote_iris* remotely mounts the *local_iris* file system `/usr` as `/local_iris/usr`)

Appendix 3: Setting Up BatchMin Communication

Use `/local_iris` as the mount point entry in `hostfiles.dat`.

4) Do not make files with following names:

```
*.m1  
*.m2  
*.out  
*.lck  
*.com  
*.dat  
*.log  
*.stp  
*.upt  
*.slp  
*.mmo  
*.grd  
mmodtmp.*
```

For more information, read section 3.9 on page 30 at the end of this appendix.

3.6 BatchMin Network Server Installation Instructions

Overview

The client-server works as follows. The client initiates contact with `inetd`, the internet super server, on the remote machine. The BatchMin remote daemon `bminrd` is then started by `inetd`. Once `bminrd` has been started, the client can send a request to the `bminrd` daemon (server). This can be thought of as a remote procedure call (similar to a procedure or function call in a program, but a request to do this is actually sent to a remote machine). The procedure executes and then a reply indicating the return status of the procedure call is sent from the server to the client. Additional information may also be passed back and forth during a request and/or during a reply. This mechanism of making requests and getting replies is how the BatchMin Network Server accomplishes its tasks, which include starting jobs, inquiring about the status of jobs, and retrieving the output of jobs.

Prior to 5.0 the underlying mechanism was socket-based. As of 5.0, RPC is used. RPC stands for Remote Procedure Call, a high-level communications paradigm which allows programmers to develop network applications while hiding many of the details of the underlying networking mechanism. It is used in many applications, among them NFS.

As of release 7.1 a new directory structure for MacroModel has been instituted. This was done to create a uniform installation directory structure for all products. Due to this, the new bminrd (BatchMin Server) is not compatible with the old one (and similarly the old one is not compatible with new-tree MacroModel installations). The daemons have the same name in 7.1 as it did in all versions 5.0 and later. Only one version may be installed at any given time.

Installation

If a non-NFS based network connection to a remote BatchMin server is desired, execute the following instructions. For each host that you wish to run BatchMin jobs on, the procedure outlined below must be followed.

- Load the distribution tape onto the machine that will run BatchMin.
- Log in as superuser (root).
- Follow the directions below for your computer architecture.

The information described below should contain the values as they are listed here. Exceptions are noted. Be aware that use of '\ ' means **continuation** of a line. Do **not** start a second line when entering this information as it will NOT work.

Conventions

Square brackets denote optional arguments that may be specified or not.

Appendix 3: Setting Up BatchMin Communication

[debug] If present, bminrd spits out **copious** output to the syslog file. On SGIs this file is typically in `/var/adm/SYSLOG` (note the capital letters), while on IBMs `syslogd` must be told to dump information to a file of your choosing since by default `syslogd` does not write to a file.

Angled brackets indicate that everything in between, including the brackets, should be replaced:

<brd> Absolute program name, e.g. `/usr/schrodinger/mmshare-v10005/bin/IRIX/bminrd`

<schrodir> Top level installation directory for Schrödinger products and the directory in which the BatchMin startup script resides. Use the full path and no environment variables (`$SCHRODINGER` is **not** correct).

<mmdatdir> Obsolete as of 7.1.

<mmexecdir> Obsolete as of 7.1.

Step1: Informing inetd

First, inform `inetd` about `bminrd` (i.e. register the `bminrd` program with the port-mapper). To do so, the files `/etc/inetd.conf` and `/etc/rpc` must be edited, and some system commands must be executed. Two sections follow—one specific to SGI workstations and one specific to IBM workstations. Other platforms have similar formats to their `inetd.conf` files.

SGI INSTRUCTIONS

1. To the `/etc/rpc` file, add:

```
bminrd      630474513
```

2. To `/etc/inetd.conf` add:

```
bminrd/1 stream rpc/tcp wait root <brd> bminrd <schrodir> [debug]
```

The `[debug]` option may be included (without `[]`) or not added. If used, it sends **copious** output to the system log, `SYSLOG`. Only run in debug mode when trying to diagnose a problem.

3. Send the hangup signal to the super-server:

```
/etc/killall -HUP inetd
```

4. Make sure that the daemon is registered by making sure that

```
rpcinfo -p
```

returns a line for `bminrd` that looks exactly like following except possibly for the port number (fourth column):

```
630474513 1 tcp 970 bminrd
```

IBM INSTRUCTIONS

There are some limitations in the file format for the `inetd.conf` file on IBM AIX systems. First, there is a limit of five arguments that a daemon can accept from `inetd`. This means that the optional `[debug]` format can not be used. To still allow debug and non-debug modes, use `bminrd` or `bminrd_debug` as the name of the process (first argument to the daemon, `argv[1]`).

Appendix 3: Setting Up BatchMin Communication

The second limitation is that the total number of characters for all the arguments is 50. Long path names will cause this limit to be exceeded, so links may need to be made in the / directory to the actual directories. Additionally, the names of the links located in / will need to be added to the file `inetd.conf`.

On IBM systems, perform the following steps:

1. As root, enter the command:

```
ln -s <schrodir> /schrodir
```

where `<schrodir>` is the top level installation directory as described earlier.

2. Add the following information to the file `/etc/inetd.conf`:

```
bminrd sunrpc_tcp tcp wait root <brd> bminrd 630474513 1 /schrodir
```

where `<brd>` is the absolute path to the `bminrd` program. This is located in `mmshare`, `$SCHRODINGER/mmshare-<X>/bin/<PLATFORM>/bminrd.`, where `<X>` is the version number of `mmshare` (e.g. v10016) and `<PLATFORM>` is the platform code (e.g. IRIX-mips3).

3. Add the following to the file `/etc/rpc:`

```
bminrd 630474513 bminrd
```

4. Get the process id of the `inetd` process by entering:

```
ps -ef | grep inetd
```

5. Tell it to reinitialize itself (send the hangup signal to it):

```
/etc/kill -HUP <pid>  
/usr/bin/inetd
```

where <pid> is the number of the process id.

6. Make sure the daemon is registered:

```
rpcinfo -p
```

This should return a line for bminrd that looks exactly like the following except possibly for the port number (fourth column):

```
630474513    1    tcp    970    bminrd
```

Step 2: Get rsh running

First, make sure that you have rsh working for the remote account you want to use. Entries in the `.rhosts` file (for bminrd) require the machine name to be the same as the one returned by the `hostname` command. Let us assume you are logged onto a machine called `chem1.chem.company.com` and want to run a job on the remote host `remote.chem.company.com`. You must give the user, for example joe on machine `chem1.chem.company.com`, access to the account on the machine `remote.chem.company.com`. This is done via the `.rhosts` file in the user's account on the remote machine. See the man pages on rsh and rhosts for more details.

Appendix 3: Setting Up BatchMin Communication

The host name used in the `.rhosts` file **must** be the same name as that returned by the `hostname` command. Thus, if one runs `hostname` on the local host, `chem1.chem.company.com` and it returns "chem1", then use this in the `.rhosts` file on the remote machine. If on the other hand it returns "chem1.chem.company.com", then use this. In these two cases your `.rhosts` entry would be "chem1 joe" or "chem1.chem.company.com joe", respectively. It is ok to have both entries in your `.rhosts`.

The `/etc/hosts.equiv` file can also be used to configure the `rbm` daemon. Just as `rsh` will work with the `/etc/hosts.equiv` file, so will `rbm`. Host names entered in this file have the same requirements as those that are used in the `.rhosts` file (as mentioned in the preceding paragraph). For more information see the man page on `/etc/hosts.equiv`.

Step 3: Try Out `rbm` and 'jump start' the daemon

The user interface for remote BatchMin jobs is `rbm` (`rbm` stands for Remote BatchMin). The format of an `rbm` command is:

```
rbm <bmin> <host> <login> <job> <info run halt get sleep wake update> [debug]
```

<bmin>=BatchMin binary/script to run

<host>=Name of remote host

<login>=User account on remote machine

<job>=Base name of BatchMin job to run (i.e. j1 if j1.com)

<info run halt get sleep wake update>=Select one action to perform; any one of these can be replaced by its first letter.

[debug]=Optional. If specified sends debugging information to stderr



Enter the following command:

```
rbm <bmin> <host> <login> mmodtmp i
```

Replace the <> parameters with the appropriate values. Note that `mmodtmp` is the name of the job which will not exist, and that the `rbm` program resides in a directory we refer to as `MMSHARE EXEC DIR`. The directory `MMSHARE EXEC DIR` is located in `$SCHRODINGER/mmshare-<X>/bin/<PLATFORM>`, where `<X>` is the version number of `mmshare` (e.g. `v100160`), and `<PLATFORM>` is the platform code (e.g. `IRIX-mips3`). You must choose the executable set that corresponds to the platform on which you are running since more than one platform's executable may be installed (IRIX, AIX, etc.) You will either need to add the `MMSHARE EXEC DIR` directory to your `PATH`, or invoke it with the full path name. Since this is the first time an `rbm` command is being run on the system, run the command, wait a few seconds and then interrupt it by hitting control-C (^-c). You should now be able to run `rbm` commands normally since the server is initialized.

Rerun the same command. It should return something like:

```
executable      : bmin
remote host     : chem1
remote login    : joe
job name        : mmodtmp
status          : Not running (Neither process nor
                 .log exist)
```

If you get results similar to those above, you have started the remote daemon and are ready to use the BatchMin Network Server. You can now submit, monitor and control jobs using `rbm`. Also, you can use Distributed BatchMin to run certain types of simulations on more than one computer. This provides a level of coarse-grain parallelism which can significantly decrease run time. For information on distributed BatchMin see Appendix 4 of the *BatchMin Reference Manual*.

If you wish to further configure your system so that jobs can be run from within the MacroModel GUI, follow the instructions listed in Step 4 below.

Step 4 (Optional): MacroModel—Accessing the Server From the Graphical User Interface

You can run remote jobs from the MacroModel GUI. MacroModel reads host information out of the `hostfiles.dat` file. Each entry consists of two lines. For a RPC-based server entry, add the following two-line entry to the `hostfiles.dat` file in the *MMSHARE DATA DIR* location (see section 3.1 on page 7 for more info on *MMSHARE DATA DIR*):

```
<hostname>  
*
```

The first line is the hostname of the machine on which you wish to run BatchMin. The second line consists of a '*', which can be followed optionally by an executable name. If none is given, then 'bmin' is assumed. If, for example, you had an IBM Power2 on which you wanted to run BatchMin, the second line might be '*bmin -ARCH pwr2'. This would allow you to launch a Power2 executable on the IBM. The remote host can now be chosen in MacroModel from the menu in the same fashion as NFS-based remote host entries and/or local host entries can be chosen (using the `Host` button in `ENRGY` submode).

Some Final Notes

First, either put the executable directory into the path of all users or make an alias. If you are the only person who is going to be using a BatchMin executable, then you can alias it as follows:

```
alias rbmin '$SCHRODINGER/mmshare-v10016/bin/IRIX-mips3/rbm bmin'
```

Note that "v10016" and "IRIX-mips3" must be replaced with values appropriate for your configuration.

If you want the flexibility to choose which BatchMin is used, enter the command:

```
alias rbmin '$SCHRODINGER/mmshare-v10016/bin/IRIX-mips3/rbm'
```

Note that "v10016" and "IRIX-mips3" must be replaced with values appropriate for your configuration.

In this latter case, users will have to specify which executable they want to use. This may be desirable when a user wishes to, for instance, run a mips3 specific executable on a host that can do so, taking advantage of the increased performance. In this case they might specify the executable as 'bmin -ARCH mips3'.

Troubleshooting

Can Not Connect To bminrd Server or rbm Command Seems To Hang

Check to see if the daemon has been started. This can be done by entering the command:

```
ps -ef |grep bminrd
```

You should see one bminrd process and it should be owned by root. If a process is not running, make sure that your entries in the `inetd.conf` file are spelled correctly, are valid file names/directories, and have the correct permissions. If the `inetd.conf` file is configured correctly, try to get bminrd running by reissuing the `rbm` command. If that does not work, try reinstalling the network server using the instructions above, beginning with Step 1.

Server Is Running But rbm Is Not Working Correctly

If you have problems, there are two debug modes which may help. The first and simplest can be invoked on the client side, through the `rbm` command, by adding the word "debug" to the end of the command. This will inform you of what `rbm` is trying to do and what it has or has not succeeded in doing.

If more information is required, then `bminrd` can be restarted to generate debug output to the `SYSLOG` file. This can be done by editing the `inetd.conf` file as described in Step 1 and reinitializing `inetd`. If `bminrd` is not running, then follow the above directions in Step 3 for 'jump starting' `bminrd`. If `bminrd` is already running, then determine the process id of `bminrd` and execute:

`kill <pid>`

After a few seconds, a `bminrd` process with a different process id should appear and the original one should disappear. If the original one disappears but no new one is started, then use the method of Step 3 to 'jump start' the daemon. If the debug daemon starts correctly, you will get output to the `SYSLOG` file. On an SGI, this will happen automatically and the file will typically be located in `/var/adm/SYSLOG`. On an IBM, you must tell the `syslogd` (the `syslog` daemon) to send this information to some file. For more information, see the man page on `syslogd` or see the `/etc/syslog.conf` file, which has further information on initializing system logging.

Be aware that in debug mode copious output is generated. **The `SYSLOG` file can grow to be very large.** Once debugging is no longer needed, restart the daemon in a non-debug mode.

Jobs Run Much More Slowly Than Local Jobs

Be very careful when following these directions. It is highly recommended that only skilled system administrators consider performing the procedure described below.

Remote BatchMin jobs (including distributed BatchMin jobs) are run in a special hidden directory in the user's remote account. This directory is called `._bmin`. Any time a client process connects to the server for a given account, the server first checks to see this directory exists. If it does not, the server creates it.

Typically a user's home account is on an NFS-mounted directory. Writing to an NFS-mounted directory is typically much slower than writing to a local disk. To increase the speed, the `._bmin` directory can be placed on a local disk. This is done as follows:

- Log into the remote machine using the account under which you will run jobs.
- Make sure no `._bmin` directory exists (note the dot and underscore in the name).
- If `._bmin` does exist, remove all of the files (using `rbm` or by letting the job finish), and then delete the directory.

Using the RPC Network Connection to Run BatchMin Manually

- Make a `._bmin` directory on a local disk.
- Issue the `"cd ~"` command to go back to your home directory.
- Create a soft link (this assumes there is a local disk with existing directory `/scr/joe/`):

```
ln -s /scr/joe/._bmin ._bmin
```

This will create a `._bmin` directory before-hand. When jobs are run after this, the directory already exists and files will actually be placed on the local disk.

WARNING: If this is done on any one machine where remote BatchMin jobs are going to be run, then this must be done on ALL remote hosts that use the same NFS-mounted home directory for that user account.

After the first machine is configured using the above steps, create a directory with the same name as the first machine on all additional systems. The soft link will already be present (assuming the account is the same NFS-mounted directory on all hosts). It is a requirement that the same local `._bmin` directory exist on each machine where the jobs will be run. Thus, if the first machine is configured to have a soft link which points to a directory called `/scr/joe/._bmin`, then all successively configured machines must also be configured this way. If not, then jobs on those hosts will not run since the soft link will point to a non-existent directory.

3.7 Using the RPC Network Connection to Run BatchMin Manually

If the BatchMin RPC-based daemon has been installed, a command line interface - **rbm**, which prior to 5.0 was called **rbmin**, provides network transparent BatchMin job control without starting MacroModel. As of release 7.1, `rbm` is located in the `MMSHARE EXEC DIR` directory (see section 3.1 on page 7 for more information). It is executed with the following arguments:

```
rbm <executable name> <host> <login> <job> <info run halt get sleep wake update>
```

Items surrounded by `<>` are required. You also must choose one of the arguments from the list at the end of the line surrounded by `<>`.

Appendix 3: Setting Up BatchMin Communication

The executable set that is specified in place of <executable> is the BatchMin executable which the daemon will try to start. It is usually `bmin` but can be any one of the BatchMins. The <host> argument is the name of the remote computer where BatchMin is to run. The <login> argument is the login id on the remote computer where the BatchMin job is to run. The <job> argument is the name of the BatchMin job.

After these arguments are specified, choose the action desired: `run` to start a job; `info` to query the status of jobs; `halt` to stop a job; `sleep` to put a job to sleep; `wake` to restart a sleeping job, or `update` to force an update of a job. For instance:

```
rbm bmin chem1 joe job1 run
```

will start a job "job1" on the computer "chem1" under the login "joe"

Wild cards can be specified with the "-". For instance:

```
rbm bmin - - - info
```

will return the status of all your jobs on the network that you started from your current login id.

An `rbm` command with wild cards can be executed by `crontab` to start or stop jobs at specific times during the day.

For more information, read the BatchMin *Force Field and Solvent Files* in section 3.9 on page 30.

3.8 Installing Additional BatchMin Network Server Services

Prior to version 5.0, users installed a service that corresponded to the BatchMin executable that they wanted to run. Typically, the default installation would run an executable called `bmin`. If a user wanted to run a different executable, say, `bmin5K750` she would have to install an additional service.

With the advent of the RPC-based BatchMin Network Server in version 5.0, this is no longer necessary. **Prior** to release 7.1, any BatchMin executable that resided in the `<mmexecdir>` directory (specified in the `inetd.conf` file) could be run by specifying the executable. That is, the second argument to the `rbm` program was the name of the BatchMin executable. So, for example, if a user knew the remote host was a MIPS2 architecture he could specify a MIPS2 binary. This would look something like:

```
rbm bmin5K750-mips2 remote.chem.company.com joe job1 run
```

For those who installed the pre 5.0 server, we recommend that the entries for this be removed from the `/etc/services` and `/etc/inetd.conf` files.

As of release 7.1, the service argument is the name of a script located in the `$SCHRODINGER` directory. Typically it is just `"bmin"`.

```
rbm bmin remote.chem.company.com joe job1 run
```

Arguments can be passed to the `bmin` startup script. Note the quotation marks:

```
rbm "bmin -ARCH mips3" remote.chem.company.com joe job1 run
```

The above command tells the server to run `"bmin"` and pass it the arguments `"-ARCH"` and `"mips3"`. The net effect of this command is that the server will try to start up a `mips3` version of BatchMin.

3.9 BatchMin Force Field, Solvent and `atom.typ` Files

BatchMin is organized to allow both user-modified and standard default force field and solvent files to be accessed easily. BatchMin will first search for force field and solvent files in the directory where it finds the MacroModel structure file (i.e. the *directory path on the first line of the command file*). We will refer to this as the local directory or the local files. (Even though this might not be local—if, for example, the first line in the `.com` file were `/usr/you/est/input.dat` but you were actually in `/usr/you`. Typically, however, it will be the same directory since there will be just a filename without a path in this first line.) As of version 7.1, BatchMin *no longer* uses the environment variable `$BATCH_ROOT`. Instead it looks for the force field and solvent files in `$MMOD_EXEC/../../../../data` (let's call this location `MMOD DATA DIR`), where `$MMOD_EXEC` is an environment variable automatically set in a startup script based on the `$SCHRODINGER` environment variable or set explicitly by the user before the startup script is invoked.

A user can put modified force field or solvent files in the local directory from which MacroModel or BatchMin is to be executed. These user force field and/or solvent files will be used when BatchMin is run. If one wishes to use the standard default files, only the `SCHRODINGER` environment variable need be set. The default area is typically the `MMOD DATA DIR` (see above for definition). Also, users who wish to use the default files should make sure that there are no local force field or solvent files as these will override the files in the default location.

In the local directory, a file can be given either the default name (e.g. `amber.fld`) or else be named after the input file name (e.g. `job.amber`, `job.water`, etc.). Files named after the input file have a higher priority than those with the default names.

When executing remote jobs via the BatchMin Network Server, `bminrd`, the order in which BatchMin will use the files are as follows. If local files are present on the local machine, they will be shipped over to the remote host and used for that job. Note you can run more than one job on the remote host and if each has its own different set of local files, then these will be shipped over to the remote host and used only for that specific job. Thus, users can have multiple remote jobs running simultaneously each with its own copy of local (i.e. user-modified) files.

If no local files are present and if an `MMOD DATA DIR` exists in the account on the remote host, this is the location BatchMin will look for the default files. If `$MMOD_EXEC` is not defined (which is OK and usually the case), then BatchMin will try to locate automatically the files based on the `<schrodir>` directory specified in the `inetd.conf` file entry.

The same conventions hold for the `atom.typ` file.

3.10 Configuring bminrd, the BatchMin Network Server, to Start Automatically at Boot Time on an SGI

On SGI machines, the BatchMin Network Server can be configured to start up automatically at boot time. However, the server must still be installed as described above.

If you are interested in doing this, please contact us for more information at help@schrodinger.com.

