

Desmond 3.1

User Manual

Desmond User Manual Copyright © 2012 Schrödinger, LLC. All rights reserved.

While care has been taken in the preparation of this publication, Schrödinger assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

BioLuminate, Canvas, CombiGlide, ConfGen, Epik, Glide, Impact, Jaguar, Liaison, LigPrep, Maestro, Phase, Prime, PrimeX, QikProp, QikFit, QikSim, QSite, SiteMap, Strike, and WaterMap are trademarks of Schrödinger, LLC. Schrödinger and MacroModel are registered trademarks of Schrödinger, LLC. MCPRO is a trademark of William L. Jorgensen. DESMOND is a trademark of D. E. Shaw Research, LLC. Desmond is used with the permission of D. E. Shaw Research. All rights reserved. This publication may contain the trademarks of other companies.

Schrödinger software includes software and libraries provided by third parties. For details of the copyrights, and terms and conditions associated with such included third party software, see the [Legal Notices](#), or use your browser to open `$$SCHRODINGER/docs/html/third_party_legal.html` (Linux OS) or `%SCHRODINGER%\docs\html\third_party_legal.html` (Windows OS).

This publication may refer to other third party software not included in or with Schrödinger software ("such other third party software"), and provide links to third party Web sites ("linked sites"). References to such other third party software or linked sites do not constitute an endorsement by Schrödinger, LLC or its affiliates. Use of such other third party software and linked sites may be subject to third party license agreements and fees. Schrödinger, LLC and its affiliates have no responsibility or liability, directly or indirectly, for such other third party software and linked sites, or for damage resulting from the use thereof. Any warranties that we make regarding Schrödinger products and services do not apply to such other third party software or linked sites, or to the interaction between, or interoperability of, Schrödinger products and services and such other third party software.

Revision A, September 2012

Contents

Document Conventions	ix
Chapter 1: Introduction	1
1.1 Installation and Configuration	2
1.2 The Maestro Interface to Desmond	3
1.3 Desmond Calculations Overview	4
1.4 Running Schrödinger Software	5
1.5 Citing Desmond in Publications	7
Chapter 2: Building a Model System	9
2.1 Adding Solvent	9
2.2 Setting Up the Boundary Box	11
2.3 Adding a Membrane	11
2.4 Using Custom Charges	13
2.5 Specifying the Force Field	14
2.6 Adding Ions	14
2.6.1 Defining an Excluded Region.....	15
2.6.2 Ion Placement.....	15
2.6.3 Adding a Salt.....	16
2.7 Running the Job	16
2.8 Rebuilding a Model System	17
2.9 Quick Setup Instructions	17
Chapter 3: Running Simulations from Maestro.....	19
3.1 Overview of the General Desmond Panels	19
3.2 Selecting a Model System	20
3.3 Minimizations	21
3.4 Molecular Dynamics Simulations	22

3.5	Simulated Annealing Simulations	24
3.6	Replica Exchange Simulations	26
3.7	Metadynamics Simulations	28
3.8	Simulations on Systems with Membranes	30
3.9	Setting Options for Desmond Simulations	31
3.9.1	The Integration Tab	32
3.9.2	The Ensemble Tab	33
3.9.3	The Minimization Tab	34
3.9.4	The Interaction Tab	35
3.9.5	The Restraints Tab	36
3.9.6	The Output Tab	37
3.9.7	The Misc Tab	38
3.10	Running a Simulation Job	39
Chapter 4: Running FEP Simulations		41
4.1	FEP Panels	41
4.2	Ligand Functional Group Mutation	42
4.3	Ring Atom Mutation	44
4.4	Protein Residue Mutation	46
4.5	Total Solvation Free Energy Calculation	48
4.6	Selecting the Environment and FEP Protocol	49
4.7	FEP Results	51
4.7.1	Relative Free-Energy Differences	51
4.7.2	Total Solvation Free Energies	52
4.7.3	FEP Analysis	53
4.8	Customizing and Restarting FEP Simulations	53
Chapter 5: Analyzing Simulations		57
5.1	Viewing Trajectories	57
5.2	Simulation Quality Analysis	61

5.3	Simulation Event Analysis	62
5.4	Replica Exchange Review	63
5.5	Metadynamics Analysis	64
Chapter 6: Running Simulations from the Command Line		67
6.1	The desmond Command	67
6.2	Running Multiple Simulations	70
6.2.1	Examples of Running MultiSim	72
6.2.2	Sample MultiSim Job (.msj) File	72
6.2.3	Treatment of Intermediate Files	75
6.3	Building a Model System	75
6.3.1	Reading the Structures	76
6.3.2	Adding a Membrane	78
6.3.3	Setting the Box Shape and Dimensions	78
6.3.4	Setting Force Field Information	79
6.3.5	Setting the Number and Location of Ions	79
6.3.6	Solvating the System	80
6.3.7	Writing the Output File	80
Chapter 7: Using VMD for Desmond Trajectories		81
7.1	Reading a CMS File and a Desmond Trajectory	81
7.2	Writing a Maestro File	82
Chapter 8: Alternate Force Field Parameters and Constraints		85
8.1	The viparr Utility	85
8.2	The build_constraints Utility	87
8.3	Input and Output Files	88
8.4	Specifying Multiple Force Fields	89
8.5	User-Defined Force Fields	90
8.6	Known Issues	91

Chapter 9: Utilities.....	93
9.1 solvate_pocket.....	93
9.1.1 Methodology	93
9.1.2 Command Syntax	94
9.1.3 Command File Syntax.....	95
9.2 manipulate_trj.py.....	98
9.3 amber_prm2cms.py	99
9.4 mold_gpcr_membrane.py.....	100
9.5 trajectory_extract_frame.py	101
9.6 desmond_restraints.py	102
9.7 rebuild_cms.py	103
9.8 Other Scripts	104
Appendix A: The multisim Utility.....	105
A.1 Running multisim	105
A.1.1 Template multisim Commands	105
A.1.2 Node Locking.....	106
A.1.3 Restarting multisim Jobs	106
A.1.4 Obtaining Information from multisim Checkpoint Files	108
A.2 The multisim File Syntax	109
A.2.1 General Keywords	111
A.2.2 Desmond-Specific Common Keywords	111
A.2.3 The restrain Keyword.....	112
A.2.4 The atom_group Keyword	113
A.2.5 The task Stage	114
A.2.6 The system_builder Stage.....	116
A.2.7 The build_geometry Stage	117
A.2.8 The assign_forcefield Stage	118
A.2.9 The simulate, replica_exchange, and lambda_hopping Stages	118
A.2.10 The minimize Stage.....	119

A.2.11	The solvate_pocket Stage	119
A.2.12	The extern Stage	121
A.2.13	The analysis Stage	123
A.2.14	The fep_analysis Stage	124
A.2.15	The vrun and fep_vrun Stages	125
A.2.16	The trim Stage	125
Appendix B: The Configuration File		127
B.1	Ark Format Syntax Summary	127
B.2	Units	128
B.3	Macros in String Values	128
B.4	Common Front-End Parameters	129
B.4.1	fep	129
B.4.1.1	fep.lambda	130
B.4.1.2	fep.i_window	130
B.4.1.3	Default Lambda Schedules	130
B.4.2	cutoff_radius	131
B.4.3	taper	131
B.4.4	coulomb_method	132
B.4.5	temperature	132
B.4.6	annealing	133
B.4.7	pressure	133
B.4.8	surface_tension	133
B.4.9	ensemble	133
B.4.10	time	134
B.4.11	elapsed_time	134
B.4.12	timestep	134
B.4.13	cpu	135
B.4.14	glue	135
B.4.15	trajectory	135
B.4.16	eneseq	136
B.4.17	ckpt	136
B.4.18	maeff_output	137

B.4.19	randomize_velocity	137
B.4.20	simbox_output	138
B.4.21	energy_group	138
B.4.22	backend	138
B.5	Parameters for Minimization	139
B.5.1	max_steps	139
B.5.2	convergence	139
B.5.3	steepest_descent_steps	139
B.5.4	num_vector	139
B.6	Parameters for Replica Exchange Simulations	140
B.6.1	replica	140
B.7	Parameters for Metadynamics Simulations	141
B.7.1	meta.....	141
B.7.2	cv	141
B.8	Parameters for vrun	142
B.8.1	vrun_frameset.....	142
Appendix C: Analyzing a Simulation from the Command Line.....		143
C.1	simulation_block_data.py	143
C.2	simulation_block_test.py	144
C.3	Simulation Block Analysis (.sba) File Syntax	144
C.4	Simulation Block Test (.sbt) File Syntax	145
References.....		147
Getting Help		151
Index.....		155

Document Conventions

In addition to the use of italics for names of documents, the font conventions that are used in this document are summarized in the table below.

Font	Example	Use
Sans serif	Project Table	Names of GUI features, such as panels, menus, menu items, buttons, and labels
Monospace	<code>\$SCHRODINGER/maestro</code>	File names, directory names, commands, environment variables, command input and output
Italic	<i>filename</i>	Text that the user must replace with a value
Sans serif uppercase	CTRL+H	Keyboard keys

Links to other locations in the current document or to other PDF documents are colored like this: [Document Conventions](#).

In descriptions of command syntax, the following UNIX conventions are used: braces { } enclose a choice of required items, square brackets [] enclose optional items, and the bar symbol | separates items in a list from which one item must be chosen. Lines of command syntax that wrap should be interpreted as a single command.

File name, path, and environment variable syntax is generally given with the UNIX conventions. To obtain the Windows conventions, replace the forward slash / with the backslash \ in path or directory names, and replace the \$ at the beginning of an environment variable with a % at each end. For example, `$SCHRODINGER/maestro` becomes `%SCHRODINGER%\maestro`.

Keyboard references are given in the Windows convention by default, with Mac equivalents in parentheses, for example CTRL+H (⌘H). Where Mac equivalents are not given, COMMAND should be read in place of CTRL. The convention CTRL-H is not used.

In this document, to *type* text means to type the required text in the specified location, and to *enter* text means to type the required text, then press the ENTER key.

References to literature sources are given in square brackets, like this: [10].

Introduction

Desmond is an explicit-solvent molecular dynamics program developed by D. E. Shaw Research. Desmond was created from scratch with an emphasis on accuracy, speed and scalability. It supports many of the most sought-after features in a modern molecular dynamics program, including:

- Highly scalable parallel execution
- Explicit solvent simulations with periodic boundary conditions using cubic, orthorhombic, and triclinic simulation boxes. Truncated octahedron and rhombic dodecahedron are supported via their triclinic analogues.
- Support for isotropic, semi-isotropic and anisotropic pressure coupling
- Smooth particle mesh Ewald method for accurate and efficient evaluation of long-range electrostatics
- NVE, NVT, NPT, NPAT, NP γ T ensembles with Berendsen, Langevin, or Nosé-Hoover thermostats, and Berendsen, Langevin, or Martyna-Tobias-Klein barostats
- Symplectic integration of the equations of motion using a multiple time step approach, RESPA
- Elimination of many sources of numerical error, permitting accurate and fast calculations using single-precision arithmetic
- Accurate implementation of constraints to eliminate high-frequency motions and thus permit larger time steps
- Exploitation of modern computer chip features to enhance speed (SIMD)
- Efficient calculation of the pressure
- Accurate checkpointing mechanism for continuing or restoring simulations
- Template-based support for widely-used force fields (using `viparr`).
- Viewing of trajectories with VMD using a Desmond plug-in.

A description of Desmond was published, along with performance data, as part of the conference proceedings of the ACM/IEEE Conference on SuperComputing 2006 (SC06) [1]. While developing Desmond, D. E. Shaw Research has introduced and extended a number of scientific algorithms, including new parallelization strategies and numerical techniques, some of which

have been published [2–5].

Problem-solving often involves using a wide range of modelling techniques, so integrating Desmond into Schrödinger's premier molecular modelling suite for drug development enhances the utility of both. Examples of such synergies include:

- The Protein Preparation Wizard, LigPrep (ligand structure) and Epik (ligand protonation state) preparation tools can be used to ensure that the structures provided to Desmond are chemically correct. Such careful system preparation often represents a crucial step prior to initiating a molecular dynamics simulation.
- Prime can be used to create homology models for use in simulations and to repair protein structures.
- Glide can be used to generate relevant poses within protein binding sites for use in simulations. Desmond in turn can be used to thermally relax, refine, and sample conformations related to the docked poses.
- Strike can be used to generate statistical models from the results of simulations.
- Desmond can be used to sample protein structures prior to performing docking calculations with Glide.
- SiteMap can be used to identify potential binding sites from simulation results.
- WaterMap analyzes specially designed Desmond simulations to characterize the thermodynamics of water in protein binding sites.

1.1 Installation and Configuration

Desmond is supported on x86 hardware under Linux and Mac OS X, and is available in both 32-bit and 64-bit versions. Detailed requirements and installation and configuration instructions are given in the [Installation Guide](#). Desmond itself is not supported on Windows, but the Maestro graphical interface is supported, and you can run jobs on remote Linux or Mac hosts from Windows.

Although Desmond can run serially, for most purposes, you will want to make use of the parallel execution capabilities. Desmond uses Open MPI for parallel execution. Before you can run jobs, however, you must add entries to the hosts file for parallel execution with Open MPI, in addition to any configuration that is needed for the hosts and the queueing system. See the [Installation Guide](#) for instructions, especially [Chapter 7](#) and [Section 7.3.3](#).

1.2 The Maestro Interface to Desmond

A number of Maestro panels have been provided to streamline the process of setting up, running and understanding the results of Desmond jobs so that you can focus on what you are studying. These panels can be used on all supported platforms to prepare and analyze jobs (Linux, Mac, and Windows).

In addition, much of the framework for running Desmond jobs has been written in Python to facilitate adaption to user-specific requirements, including the automation of larger and more specific workflows.

System Builder panel

- Constructs systems suitable for simulation using periodic boundary conditions
- Bulk solvent and membrane environments supported
- Solvation and neutralization largely automated yet customizable
- Seamless force-field parameter assignment
- Can be run locally on Windows

Minimization panel

Molecular Dynamics panel

Simulated Annealing panel

Replica Exchange panel

Metadynamics panel

- Desmond job launching
- Ability to intuitively see and adjust the key parameters used by the Desmond program for both minimization and dynamics calculations
- Default parameters are suitable for many simulations
- Intelligent coupling of related settings
- Access to both minimization and simulation settings
- Easy simulation continuation and restoration
- Optional automated relaxation and equilibration procedures

Ligand Functional Group Mutation by FEP panel

Ring Atom Mutation by FEP panel

Protein Residue Mutation by FEP panel

Total Free Energy by FEP panel

- Easy to use with a focus on the real problem of interest rather than the details of the calculation
- Support for absolute and relative solvation free energy calculations
- Support for relative binding free energy calculations
- Restarting and customization of FEP jobs via FEP panel

Trajectory Viewer panel

- Integrated into Maestro
- Comprehensive speed controls
- Replication of primary simulation box for viewing
- Easy creation of images and movies.

Simulation Quality Analysis panel

- Intuitive tool for examining certain markers of simulation quality

Simulation Event Analysis panel

- Intuitive tool for investigating what happened during a simulation

The use of these panels is described in subsequent chapters of this manual.

Most Desmond-related tools are available from the Desmond submenu of the Applications menu in Maestro. The exceptions are the trajectory viewer, which is opened from the Project Table using the output entry for a job. Desmond panels can also be opened from Tasks → Molecular Dynamics.

1.3 Desmond Calculations Overview

Desmond jobs should be started from well-prepared structures. For proteins it is recommended that the protein be prepared with the Protein Preparation Wizard (see the [Protein Preparation Guide](#) for details). For other types of molecules, such as ligands, the molecule should have a fairly good Lewis structure (although there are some built-in capabilities for adjusting incorrect or non-optimal Lewis structures).

If you have MacroModel you can perform a quick check on the structure by performing a Current Energy calculation (available from the MacroModel submenu of the Applications menu) using the OPLS_2005 force field with the Solvent set to None. If that calculation succeeds it is almost certain that Desmond and its associated tools will be able to work with this structure as well. If the structure is problematic Maestro and MacroModel often provide useful diagnostics for what might be wrong.

Performing a study based on a Desmond molecular dynamics simulation usually involves a number of stages, including simulation setup, relaxing the system (this could just be a minimization), running the simulation, viewing trajectories, and analyzing the results. Simulation setup is described in [Chapter 2](#), the basic Desmond minimization, molecular dynamics, simulated annealing, replica exchange, and metadynamics tasks are described in [Chapter 3](#). Simplified FEP setup for relative binding and solvation free energies and absolute solvation free energies is described in [Chapter 4](#), along with restarting and customizing FEP simulations.

Examining the results, including viewing a trajectory, and analysis of results, is described in [Chapter 5](#).

FEP jobs are handled differently due to the complexity of the calculations and the fact that the overall goal for an FEP job is to produce one number: the free energy change. FEP jobs are supported for specific types of calculations, using automated procedures that differ from those used for individual, general purpose Desmond simulations.

The basic outline of a Desmond simulation as run from Maestro is as follows:

1. Import the structure file for the system of interest into Maestro.
2. Prepare the structure for simulation with the Protein Preparation Wizard. This step involves removing ions and molecules (which are artifacts of crystallization), setting correct bond orders, adding hydrogens, filling in missing side chains or whole residues as necessary, reorienting various groups and varying residue protonation states to optimize the hydrogen bonding network, and then checking the structure carefully.
3. If your system is a membrane protein, embed the protein in the membrane. This step and the next two steps are performed in the System Builder panel.
4. Generate a solvated system for simulation.
5. Distribute positive or negative counter ions to neutralize the system, and introduce additional ions to set the desired ionic strength (when necessary).
6. Relax the system either by minimization or by selecting the panel option to relax the model system before simulation.
7. Set the simulation parameters in one of the general Desmond panels, for molecular dynamics, simulated annealing, or replica exchange.
8. Run the simulation.
9. Analyze your results using the Trajectory Viewer and other analysis tools.

1.4 Running Schrödinger Software

Schrödinger applications can be started from a graphical interface or from the command line. The software writes input and output files to a directory (folder) which is termed the *working directory*. If you run applications from the command line, the directory from which you run the application is the working directory for the job.

Linux:

To run any Schrödinger program on a Linux platform, or start a Schrödinger job on a remote host from a Linux platform, you must first set the `SCHRODINGER` environment variable to the installation directory for your Schrödinger software. To set this variable, enter the following command at a shell prompt:

```
csh/tcsh:      setenv SCHRODINGER installation-directory
bash/ksh:      export SCHRODINGER=installation-directory
```

Once you have set the `SCHRODINGER` environment variable, you can run programs and utilities with the following commands:

```
$SCHRODINGER/program &
$SCHRODINGER/utilities/utility &
```

You can start the Maestro interface with the following command:

```
$SCHRODINGER/maestro &
```

It is usually a good idea to change to the desired working directory before starting Maestro. This directory then becomes Maestro's working directory.

Windows:

The primary way of running Schrödinger applications on a Windows platform is from a graphical interface. To start the Maestro interface, double-click on the Maestro icon, on a Maestro project, or on a structure file; or choose **Start → All Programs → Schrodinger-2012 > Maestro**. You do not need to make any settings before starting Maestro or running programs. The default working directory is the Schrodinger folder in your documents folder (Documents on Windows 7/Vista, My Documents on XP).

If you want to run applications from the command line, you can do so in one of the shells that are provided with the installation and that have the Schrödinger environment set up:

- Schrödinger Command Prompt—DOS shell.
- Schrödinger Power Shell—Windows Power Shell (if available).

You can open these shells from **Start → All Programs → Schrodinger-2012**. You do not need to include the path to a program or utility when you type the command to run it. If you want access to Unix-style utilities (such as `awk`, `grep`, and `sed`), preface the commands with `sh`, or type `sh` in either of these shells to start a Unix-style shell.

Mac:

The primary way of running Schrödinger software on a Mac is from a graphical interface. To start the Maestro interface, click its icon on the dock. If there is no Maestro icon on the dock, you can put one there by dragging it from the SchrodingerSuite2012 folder in your Applications folder. This folder contains icons for all the available interfaces. The default working directory is the Schrodinger folder in your Documents folder (\$HOME/Documents/Schrodinger).

Running software from the command line is similar to Linux—open a terminal window and run the program. You can also start Maestro from the command line in the same way as on Linux. The default working directory is then the directory from which you start Maestro. You do not need to set the SCHRODINGER environment variable, as this is set in your default environment on installation. If you need to set any other variables, use the command

```
defaults write ~/.MacOSX/environment variable "value"
```

1.5 Citing Desmond in Publications

The use of this product should be acknowledged in publications as:

Desmond Molecular Dynamics System, version 3.1, D. E. Shaw Research, New York, NY, 2012. Maestro-Desmond Interoperability Tools, version 3.1, Schrödinger, New York, NY, 2012.

Please also include a reference to the following paper:

Kevin J. Bowers, Edmond Chow, Huafeng Xu, Ron O. Dror, Michael P. Eastwood, Brent A. Gregersen, John L. Klepeis, Istvan Kolossvary, Mark A. Moraes, Federico D. Sacerdoti, John K. Salmon, Yibing Shan, and David E. Shaw, “Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters,” Proceedings of the ACM/IEEE Conference on Supercomputing (SC06), Tampa, Florida, November 11-17, 2006.

Building a Model System

Performing simulations on aqueous biological systems requires the preparation of biological molecules such as proteins and ligands, addition of counter ions to neutralize the system, selection of simulation box size, solvation of the solutes using explicit solvent molecules, and alignment of proteins to a membrane bilayer (if used). This procedure is often tedious if it has to be performed manually. Tools for all these tasks are provided with Maestro.

Protein and ligand structures used in a Desmond simulation must be complete all-atom 3D structures with a reasonable geometry. The preparation of protein and ligand structures for use in a simulation can be done with the Protein Preparation Wizard and LigPrep. The Protein Preparation Wizard corrects structural defects, adds hydrogen atoms, assigns bond orders, and can selectively assign tautomerization and ionization states, and optimize the hydrogen bonding network. For more information, see the [Protein Preparation Guide](#). LigPrep performs 2D-to-3D conversion if necessary, adds hydrogen atoms, generates tautomers, ionization states, ring conformations, and stereoisomers, as requested, and produces minimized 3D structures. For more information, see the [LigPrep User Manual](#).

Once you have prepared the protein and ligand structures, you can proceed to the remaining tasks in building a model system that can include proteins, ligands, explicit solvent, a membrane, and counter ions. The System Builder automates this process and significantly reduces the effort required. You can set up and run a System Builder job from the System Builder panel, or from the command line. See [Section 6.3 on page 75](#) for information on running the System Builder from the command line.

To open the System Builder panel, do one of the following:

- Choose Applications → Desmond → System Builder
- Choose Tasks → Molecular Dynamics → System Setup

Before you start working in this panel, display the solutes in the Workspace.

2.1 Adding Solvent

The solvation model is selected in the Solvation tab. You can choose from a set of predefined solvent models, or specify a custom solvent model:

- **None**—Do not use a solvent. This option allows you to run a simulation on a pure liquid, for example, or in vacuum (with a sufficiently large box).

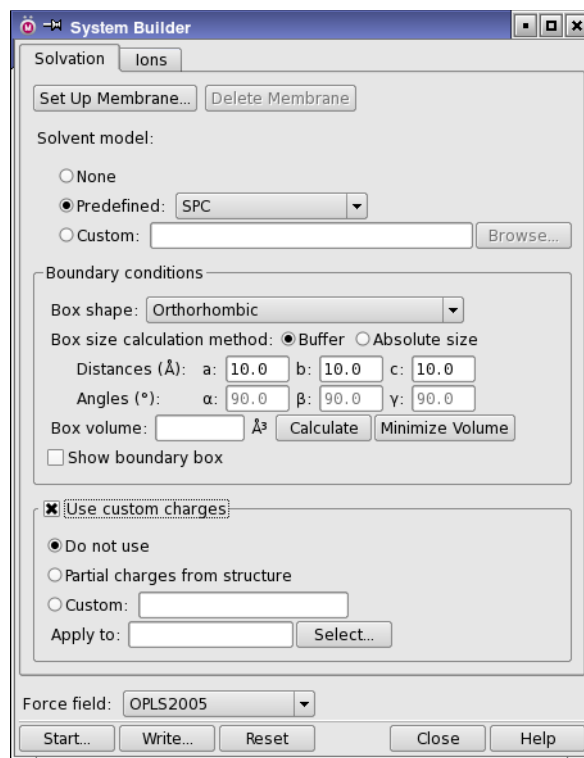


Figure 2.1. The Solvation tab of the System Builder panel.

- **Predefined**—Use one of the predefined solvent models, which you can select from the option menu. The models include four water models, SPC, TIP3P, TIP4P, and TIP4PEW, and three organic solvents, methanol, octanol, and dimethyl sulfoxide (DMSO).
- **Custom**—Import a custom solvent system from file. Enter the solvent system file name in the text box, or click **Browse** and navigate to the solvent system file in the file selector that is displayed.

The solvent is placed by replicating “boxes” of solvent molecules and deleting molecules whose center of mass lies outside the periodic box boundary, and molecules that are inside or have significant overlap with the solute or the membrane (if one is used).

2.2 Setting Up the Boundary Box

The periodic boundary conditions are set up by specifying the shape and size of the repeating unit, or box, which you can do in the Solvation tab.

To set up the box, first choose the shape from the Box shape option menu. Three basic shapes are provided: Cubic, Orthorhombic, and Triclinic. As special cases of the triclinic box shape, three other shapes are supported: Truncated octahedron, Rhombic dodecahedron xy-square, and Rhombic dodecahedron xy-hexagon.

When you have chosen the box shape, you can choose whether to specify the size of the box in terms of a buffer distance or as an absolute size, by selecting one of the Box size calculation method options:

- **Buffer**—The simulation box size is calculated by using the given buffer distance between the solute structures and the simulation box boundary.
- **Absolute size**—Specify the lengths of the sides of the simulation box size (and angles if necessary).

Having chosen a method, you can specify the distances and angles in the Distances and Angles text boxes. The text boxes that are available depend on the box shape. For all choices except a truncated octahedron, the box can be displayed in the Workspace by selecting Show boundary box.

If you want to calculate the volume of the box that encompasses the solutes, click Calculate. The volume is displayed in the Box volume text box. To minimize the volume of the box, click Minimize Volume. The solutes are reoriented so that the box volume is minimized.

2.3 Adding a Membrane

A membrane can be added to the system using the Set Up Membrane dialog box, which you open by clicking Set Up Membrane in the Solvation tab. This dialog box allows you to select and position the membrane; the actual membrane is added when the system builder job is run.

There are four predefined membranes, DPPC, DMPC, POPC, and POPE, which you can choose by selecting Predefined, and choosing the membrane from the option menu. The temperature at which the membrane patch was preequilibrated is given in parentheses after the membrane name. Because DPPC has a gel transition temperature around 313 K, the recommended minimum simulation temperature is also 325 K.

If you want to position a custom membrane, select Custom, and enter the name of the Maestro file containing the membrane model in the text box, or click Browse and navigate to the file.

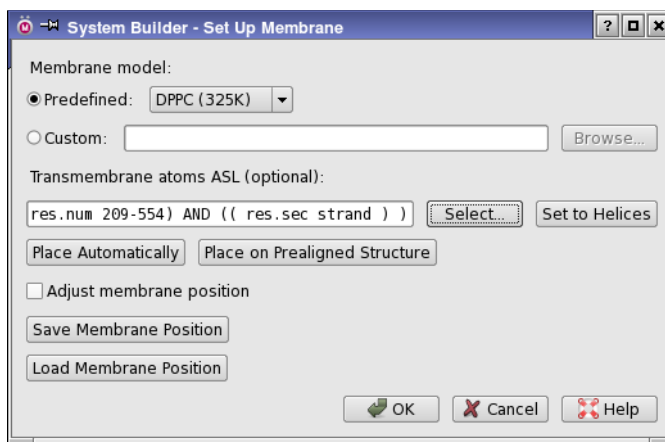


Figure 2.2. The Set Up Membrane panel.

If you have an existing membrane in a project entry that you want to use for the current model system, you can load it by selecting the entry and clicking Load Membrane Position from Selected Entry. The membrane from this entry is then used for the model system you are building.

When you click Place Automatically, the membrane position is determined according to the information available, as follows:

- If you have a protein from the OPM database (<http://opm.phar.umich.edu>), the membrane is placed using the information provided with the protein.
- Otherwise the surface of the membrane is placed perpendicular to the longest axis of the protein.
- If transmembrane atoms are defined, they are placed inside the membrane. Placement of transmembrane atoms inside the membrane takes precedence over placement perpendicular to the longest axis.

To define the transmembrane atoms, you can do one of the following:

- Click Select, and use the Atom Selection dialog box to select the desired atoms. For more information on this dialog box, see [Section 6.5](#) of the *Maestro User Manual*.
- Click Set to Helices. This button sets the ASL expression to `res.sec helix`, which selects the residues whose secondary structure assignment is “helix”. The protein must have a secondary structure assignment before you use this button. To do the assignment, choose Tools → Assign Secondary Structure in the main window.

If you have a protein that is prealigned, you can click **Place** on **Prealigned Structure** to place the membrane. The membrane is positioned symmetrically about the coordinate origin so that its surfaces are parallel to the xy plane (perpendicular to the z axis). This means that the protein must be aligned accordingly.

When you have placed the membrane, a representation of the membrane is displayed in the **Workspace**. The representation consists of two red slabs for the surfaces, with a yellow line perpendicular to the slab planes. After the membrane has been placed, you can adjust its orientation by selecting **Adjust membrane position**, and rotating the membrane. The actual membrane molecules are placed when the system builder job is run. The molecules are placed by replication of a membrane segment and deletion of molecules whose center of mass lies outside the periodic box boundaries. Molecules that are inside the solute or have significant overlap with it are removed to accommodate the solute.

If you click **Place Automatically** after adjusting the membrane, the membrane is returned to its default position and orientation.

The membrane position and orientation can be stored in **Project Table** entries, by selecting the entries in the **Project Table**, and clicking **Save Membrane Position to Selected Entries**. This enables the membrane position and orientation to be loaded at a later time by selecting the entry and clicking **Load Membrane Position from Selected Entry**.

It can be difficult to set up GPCR systems properly. The `mol_d_gpcr_membrane.py` script can be used to swap your GPCR protein into a system that has already been constructed for a related protein. Templates for a number of GPCR systems are available. See [Section 9.4 on page 100](#) for details.

2.4 Using Custom Charges

If you want to use partial charges from a source other than the force field, you can do so by selecting **Use custom charges** in the **Solvation** tab. You can then choose to use the partial charges from the structure, or enter the name of the property that defines the custom charges in the **Custom** text box. The property name is the internal name, which should start with `r_` (i.e. a real-valued property). For example, the property `r_j_ESP_Charges` selects Jaguar-generated ESP charges.

When you have selected the property, click **Select** to select the atoms for which these charges are to be used. There is no default. The selection is made in the **Atom Selection** dialog box, which is described in detail in [Section 6.5](#) of the *Maestro User Manual*. If the property you chose has values only for some of the atoms (e.g. the ligand), you can select these atoms by specifying the entire range of values. Atoms that do not have a value for the property will not be selected.



Figure 2.3. The Ions tab of the System Builder panel.

2.5 Specifying the Force Field

The force field is specified from the choices on the Force field option menu. The default choice is OPLS_2005. To specify other force fields, you can run the utility `viparr`—see [Chapter 8](#).

2.6 Adding Ions

It is usually desirable to have an electrically neutral system for simulation (though not strictly necessary, as Desmond applies a uniform background charge distribution to neutralize the system in the Ewald summation). You can choose to add ions to neutralize the system in the Ion placement section of the Ions tab. The system can also be set up in a salt solution rather than a pure solvent in the Add salt section of the Ions tab. To limit the locations in which ions can be placed, you can define regions from which ions are excluded, in the Exclude region section of the Ions tab.

2.6.1 Defining an Excluded Region

To define an excluded region, click **Select** in the **Excluded region** section of the **Ions** tab, and use the **Atom Selection** dialog box to select the desired set of residues. You should select residues that are within or near the binding site. When ions are placed, they will not be placed near these residues. The residues that you select are colored blue and rendered in CPK. See [Section 6.5](#) of the *Maestro User Manual* for more information on the **Atom Selection** dialog box. The region is defined by the distance in the **Exclude ion and salt placement within** text box. Ions will not be placed within the specified distance of the selected atoms.

2.6.2 Ion Placement

Ions are placed in the solvent according to your selection in the **Ion placement** section of the **Ions** tab. Each ion replaces a solvent molecule. You can, of course, choose not to add ions, by selecting **None**.

If you select **Neutralize**, the minimum number of sodium or chloride ions required to balance the system charge is placed randomly in the solvent.

If you select **Add**, you can choose the ion type from the option menu and enter the number of ions to add (which need not neutralize the system). The option menu only displays ions that are opposite in charge to that of the system. Ions are not placed in the excluded regions.

Instead of placing ions automatically, at random, you can locate and select suitable regions for ions to be placed. Usually these regions are near residues that have the same charge as the system charge and are not near the active site. You can define these regions in the **Advanced Ion Placement** dialog box, which you open by clicking **Advanced Ion Placement** in the **Ions** tab.

To place the ions, you must identify suitable candidate residues. When you click **Candidates**, the **Candidates** table is populated with a list of residues in regions that have not been excluded and have the same charge as the overall charge of the system. These residues are colored red and rendered in CPK. Ions are placed near the residues that you select in the table, replacing the closest solvent molecule to the average position of the atoms in the residue. The number of ions placed (initially 0), along with the number of ions remaining to be placed and the total number of candidate residues are displayed above the table.

You can add candidates to the table by clicking **Select**, and selecting the residues in the **Atom Selection** dialog box. When you click **OK** in the dialog box, the residues are added to the table, and can be selected along with the automatically located residues. To clear the table, click **Reset**.

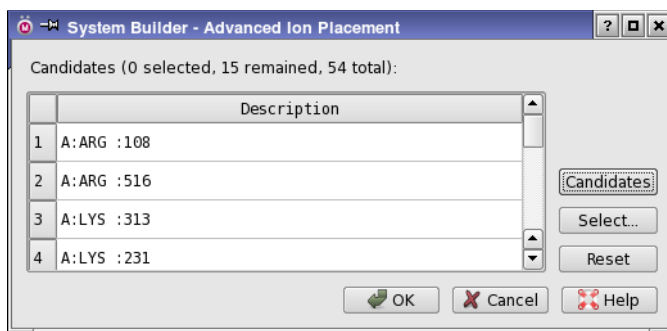


Figure 2.4. The Advanced Ion Placement dialog box.

When the system builder job is run, ions that are placed using the Advanced Ion Placement dialog box are placed first. Once these ions are placed, random placement is performed to place any remaining ions that are needed to neutralized the system or complete the number of ions selected for placement in the Add text box.

2.6.3 Adding a Salt

Adding a salt is relatively simple. To do so, first select the Add salt option. The controls in the Add salt section are then activated, and you can enter the salt concentration, in mol dm^{-3} , and select the desired ions. If you select multiply charged ions, the concentration is taken from the empirical formula for the salt. For example, for MgCl_2 the concentration of Mg^{2+} would be the specified concentration and the concentration of Cl^- would be twice the specified concentration. A value of 0.15M is approximately the physiological concentration of monovalent ions.

When the salt ions are placed, they are randomly distributed in the solvent, and replace solvent molecules. They are not placed in the excluded region defined in the Exclude region section.

2.7 Running the Job

When you have finished making settings, you can set up and start the job immediately, or write out the input file and run the job from the command line.

To set up and run the job, click **Start**. The **Start** dialog box opens, allowing you to name the job, choose the host and set the user name (if necessary). System Builder jobs do not usually take more than a few minutes, so you can run the job locally on Linux systems (but not on Windows). You can also choose whether to incorporate the output CMS file back into the Maestro project, by choosing **Append new entries** from the **Incorporate** option menu. This is useful if you want to continue on to set up a simulation in Maestro. If you choose **Do not incorporate**, the CMS file is placed in the current working directory, but is not added to the project.

If you want to run the job from the command line, click **Write**. The **Write** dialog box opens, in which you can specify a name and then write the file. The name is used to construct the file names, by adding the appropriate extension.

2.8 Rebuilding a Model System

If you want to modify an existing model system (full system CT), and run a simulation on it, you do not need to completely rebuild the system in the **System Builder** panel. Instead, you can simply regenerate the model system. To do so, include the full model system in the **Workspace**, then choose **Applications** → **Desmond** → **Model System Regeneration**. A **Start** dialog box opens, so you can specify the usual job parameters.

If you want to rebuild a model system from the command line, you should run one job from **Maestro**, and then edit the `multisim` input file (`.msj`) for other model systems.

2.9 Quick Setup Instructions

The sets of instructions below take you through the simplest setup procedures. It is assumed that you have imported the prepared protein and ligand structures into **Maestro**, and displayed them in the **Workspace**.

To add solvent:

1. Select **Predefined** for the **Solvent model** option, and choose a model from the option menu.
2. Choose a box shape.
3. Choose a box size calculation method—**Buffer** for adding a buffer region to the solutes, or **Absolute size** for specifying the actual box size.
4. Enter buffer distances or side lengths in the available text boxes.
5. Enter angles if you selected **Triclinic** for the box shape.

To add ions:

1. In the **Ions** tab, choose an option for the addition of ions.
2. If you selected **Add**, enter the number of ions to add in the text box.
3. Choose an ion type from the option menu.
4. If the solvent is intended to be a salt solution, select **Add salt**.
5. Enter the desired salt concentration in the **Salt concentration** text box.

6. Choose positive and negative ion types from the Salt positive ion and Salt negative ion option menus.

To add a membrane:

1. Click Add Membrane in the Solvation tab.
2. In the Membrane tab, select Predefined for the membrane model, and choose a membrane type from the option menu.
3. Click Place Automatically.
4. Select Adjust membrane position and adjust the orientation of the membrane in the Workspace.
5. Click OK.

Click Start to run the job or click Write to write the input file.

Running Simulations from Maestro

The general Desmond panels enable you to set up and run the main tasks available with Desmond: molecular dynamics, minimization, simulated annealing, and replica exchange jobs. The panels are designed to make setting up these types of jobs as easy as possible, and provide the most common simulation controls. The default values provided in the panels represent a good balance between accuracy and performance, and are adequate for most jobs without change. For more control over the simulation parameters, you can make settings in the Advanced Options dialog box, which is described in [Section 3.9 on page 31](#).

A much more automated approach is provided for FEP simulations of binding and solvation free energies in four specialized panels, Ligand Functional Group Mutation by FEP, Ring Atom Mutation by FEP, Protein Residue Mutation by FEP, and Total Free Energy by FEP, for which a model system and the additional parameters are set up automatically. These panels, and the FEP panel for restarting and customizing these jobs, are described in [Chapter 4](#).

In addition to setting up simulations, you can use the general panels to restart a simulation from a checkpoint file as generated by a previously interrupted simulation.

All jobs run from these panels require a model system to be built first, in the System Builder panel—see [Chapter 2](#) for details.

Desmond simulations can also be run from the command line—see [Chapter 6](#).

3.1 Overview of the General Desmond Panels

The general Desmond panels have two main sections: Model system, in which the model system is chosen, and Simulation, in which the parameters for the task are set up (or Minimization for minimization tasks). The controls in the second section depend on the panel. Specifying a model system is described in [Section 3.2 on page 20](#), and the various tasks are described in the subsequent sections.

At the bottom of the panel are the action buttons for the job:

- **Start**—Start the job. Opens the Start dialog box to set job parameters and submit the job for execution. See [Section 3.10 on page 39](#) for details. A general description of this dialog box and its features is given in [Section 2.2](#) of the *Job Control Guide*.
- **Read**—Read a configuration (.cfg) file, to set up the simulation. Opens a file selector in which you can navigate to the desired file.

- **Write**—Write the input files for the job but do not start it. Opens a dialog box in which you can provide the job name, which is used to name the files. The job can be run from the command line, as described in [Chapter 6](#).
- **Reset**—Reset the values in the panel to their defaults.

To run a job:

1. Choose the task from the Desmond submenu of the Applications menu.
2. Specify the model system, either by loading it from the Maestro Workspace or importing it from a file.
3. Adjust the simulation parameters as necessary.

For parameters that are not available in the main panel, open the Advanced Options dialog box.
4. Click Start.
5. Set the job parameters in the Start dialog box, and click Start to run the job.

To restart a molecular dynamics job:

1. Import the checkpoint file generated by the interrupted simulation.

The default name of this file is *jobname.cpt*.

When the checkpoint file is imported, the Molecular Dynamics panel enters a read-only state, in which most of the controls are set by the information read and cannot be changed.

2. Adjust the total simulation time if necessary.
3. Click Start.
4. Set the job parameters in the Start dialog box, and click Start to run the job.

3.2 Selecting a Model System

In the Model system section, you select the model system that you will use for the simulation. A valid model system for simulations must contain both the coordinates of the particles and the force field parameters. In the case of FEP simulations, the model system should also contain additional FEP-specific parameters. A model system file normally has the *.cms* extension.

There are two options on the option menu, and the tools in this section depend on which option you choose.

- **Load from Workspace**—Load the model system from the Workspace. The Workspace must contain a model system that has already been prepared with the System Builder panel. When you choose this item, the Load button is displayed, which you click to load the model system from the Workspace.
- **Import from file**—Import the model system from a file. You can choose to import a model system file (.cms) or a checkpoint file (.cpt).

If you import a model system file, it must contain a model system that has already been prepared with the System Builder panel. When the file is imported, a message about the system is displayed below the option menu.

If you import a checkpoint file, most of the panel controls are unavailable. The purpose of the checkpoint file is to restart an interrupted simulation, so the parameters of the simulation cannot be altered. You can change the total simulation time, and then start the job.

3.3 Minimizations

Minimization jobs relax the system into a local energy minimum. The model system is minimized using a hybrid method of the steepest decent and the limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) algorithms. This task is set up in the Minimization panel, which you open by choosing Applications → Desmond → Minimization in the main window.

There are only two generally useful parameters for this task:

- **Maximum iterations**—Enter the maximum number of iterations in this text box, or use the arrow buttons to change the maximum number of iterations in steps of 10.

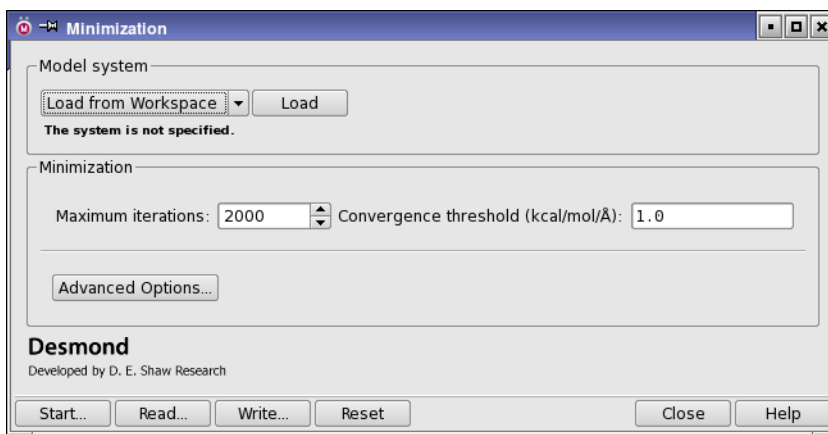


Figure 3.1. The Minimization panel.

- Convergence Threshold—Enter the convergence threshold for the gradient in units of $\text{kcal mol}^{-1} \text{\AA}^{-1}$.

3.4 Molecular Dynamics Simulations

Molecular dynamics jobs simulate the Newtonian dynamics of the model system, producing a trajectory of the particle coordinates, velocities, and energies, on which statistical analyses can be performed to derive properties of interest about the model system. The molecular dynamics task performs a single MD simulation under the chosen ensemble condition for a given model system, generating simulation data for post-simulation analyses.

This task is set up in the Molecular Dynamics panel, which you open by choosing Applications → Desmond → Molecular Dynamics in the main window.

The controls at the top of the Simulation section allow you to specify the simulation time in ns and the recording interval in ps for the energy and for the trajectory.

For the recording intervals you can enter a value in ps in the text boxes. The values are rounded to an integer multiple of the far time step size. This time step size is set in the Integration tab of the Advanced Options dialog box, in the RESPA integrator section.

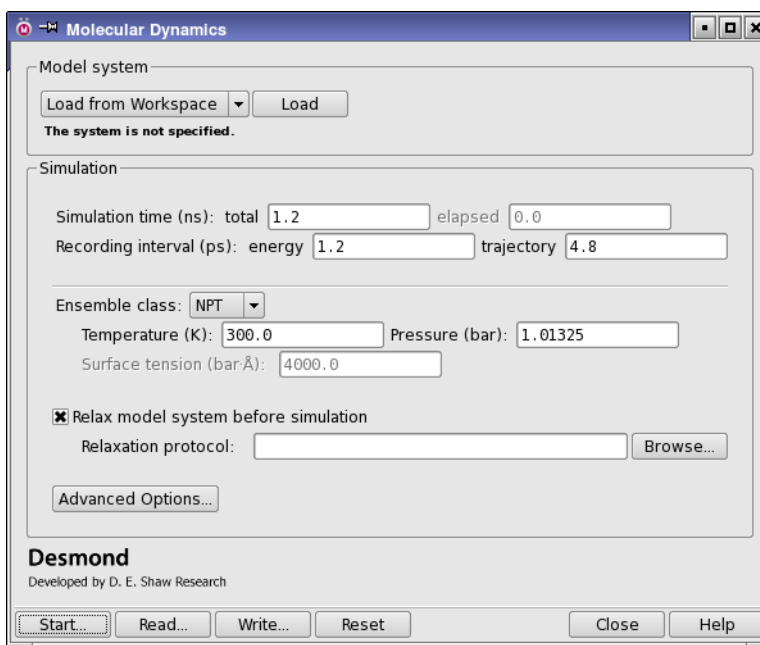


Figure 3.2. The Molecular Dynamics panel.

The controls in the lower part of the Simulation section allow you to choose the ensemble class, from NVE, NVT, NPT, NPAT, and NPγT, to set the temperature (except for NVE) and the pressure (except for NVE and NVT), and set the surface tension (NPγT only). It also allows you to relax the model system before performing the simulation, and choose the protocol for the relaxation.

When Relax model system before simulation is selected, a series of minimizations and short molecular dynamics simulations are performed to relax the model system before performing the simulation you set up. This option is selected by default, and a default protocol is used. Usually, if the model system was just created from the System Builder panel, it needs to be relaxed; if the model system has been relaxed before, it does not need to be relaxed again. As an alternative you can run a minimization before the molecular dynamics calculation.

The stages in the default relaxation process for the NPT ensemble are:

1. Minimize with the solute restrained
2. Minimize without restraints
3. Simulate in the NVT ensemble using a Berendsen thermostat with:
 - a simulation time of 12ps
 - a temperature of 10K
 - a fast temperature relaxation constant
 - velocity resampling every 1ps
 - non-hydrogen solute atoms restrained
4. Simulate in the NPT ensemble using a Berendsen thermostat and a Berendsen barostat with:
 - a simulation time of 12ps
 - a temperature of 10K and a pressure of 1 atm
 - a fast temperature relaxation constant
 - a slow pressure relaxation constant
 - velocity resampling every 1ps
 - non-hydrogen solute atoms restrained
5. Simulate in the NPT ensemble using a Berendsen thermostat and a Berendsen barostat with:
 - a simulation time of 24ps
 - a temperature of 300K and a pressure of 1 atm
 - a fast temperature relaxation constant
 - a slow pressure relaxation constant
 - velocity resampling every 1ps
 - non-hydrogen solute atoms restrained

6. Simulate in the NPT ensemble using a Berendsen thermostat and a Berendsen barostat with:

- a simulation time of 24ps
- a temperature of 300K and a pressure of 1 atm
- a fast temperature relaxation constant
- a normal pressure relaxation constant

This protocol is used for the NPAT and NPγT ensembles as well. A similar protocol is used for the NVT ensemble.

The protocol files can be found in `$SCHRODINGER/mmshare-vversion/data/desmond`. The procedure follows a similar pattern as for NPT. If you are relaxing a protein in a membrane, we strongly recommend using the `desmond_membrane_relax.ms` protocol from that directory. This protocol can take a while to run, as it involves about 1.2 ns of simulation. For more information, see [Section 3.8 on page 30](#).

If you want to modify the protocol, you can copy these files and edit them. To make use of the modified protocol, click **Browse** and navigate to the new protocol file, which has a `.ms` extension. The file name is then listed in the **Relaxation protocol** text box.

When the simulation finishes, the output structure file (`.cms`) is written to disk and incorporated into the Maestro project. In addition, a new trajectory directory is created, called `jobname_trj` by default. Checkpoint files are written during the simulation, but are not written during the relaxation process.

3.5 Simulated Annealing Simulations

Simulated annealing methods use a temperature program rather than a single temperature for the simulation. A temperature program is a series of times and target temperatures. The temperature is linearly interpolated as a function of time between adjacent target temperatures and is controlled by a thermostat.

One of the predominant strategies used is to raise the temperature to a high value one or more times before relaxing the system to the desired temperature. The goal is to permit the system to relax out of an initial state that corresponds to a high energy potential minimum into a lower state by crossing barriers in the free-energy landscape, which is achieved more effectively during the periods of elevated temperatures. The default temperature program in the **Simulated Annealing** panel falls into this category.

Another common use for simulated annealing is to perform an effective minimization with some relaxation of the system by slowly decreasing the temperature down to very low temperatures. This slow cooling should permit at least some shifts from higher energy minima to lower minima in the energy landscape.

Simulated Annealing

Model system

Load from Workspace Load

The system is not specified.

Simulation

Schedule of reference temperature change:

Number of stages: 6

	1	2	3	4	5	6
Time (ps)	30	100	200	300	500	1000
Temperature (K)	10	100	300	400	400	300

* Temperature is linearly interpolated between two adjacent time points in the schedule.

Simulation time (ns): total 1.2 elapsed 0.0

Recording interval (ps): energy 1.2 trajectory 4.8

Ensemble class: NVT

Pressure (bar): 1.01325

Surface tension (bar.Å): 4000.0

☒ Relax model system before simulation

Relaxation protocol: Browse...

Advanced Options...

Desmond
Developed by D. E. Shaw Research

Start... Read... Write... Reset Close Help

Figure 3.3. The Simulated Annealing panel.

Simulated annealing calculations can be set up and run from the Simulated Annealing panel, which you open by choosing Applications → Desmond → Simulated Annealing.

In the Simulation section, you can make settings for the simulated annealing job. The settings for the simulation time, recording interval, ensemble class and model system relaxation are the same as for a molecular dynamics simulation, and are described in [Section 3.4 on page 22](#). The main specific task for simulated annealing is to provide information on the stages by providing a schedule of reference temperature changes.

The number of stages is set in the Number of stages text box. When a value has been entered, the table below is adjusted to display text boxes for each stage. The stages are indexed from 0. For each stage you can specify a starting time in the Time text box, and a starting temperature in the Temperature text box. The temperature is linearly interpolated between adjacent time points. The last stage runs until the specified total simulation time.

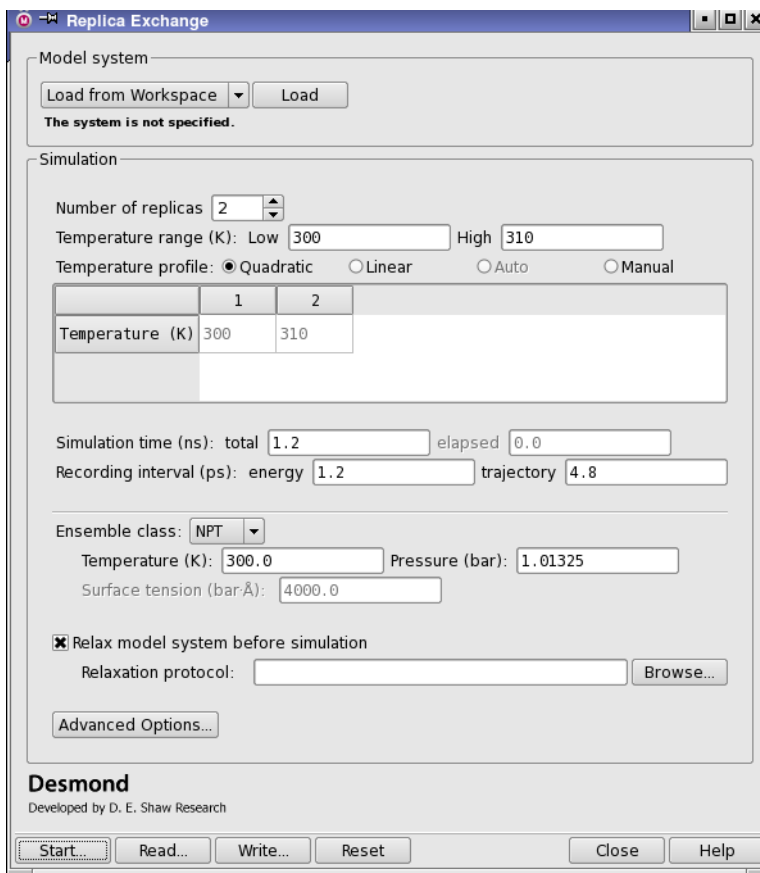


Figure 3.4. The Replica Exchange panel.

3.6 Replica Exchange Simulations

Many molecular systems have conformations that are separated by significant free energy barriers. It can be difficult to sample such conformations if they differ by concerted or collective shifts of many atoms. This commonly occurs in protein-ligand complexes. Random methods such as Monte Carlo conformational searches have trouble generating such collective changes, while thermal methods such as molecular dynamics have trouble surmounting the free energy barriers. Replica exchange simulations [41, 42] attempt to tackle this problem by allowing the system to spend some time at elevated temperatures in addition to the temperature of interest. Time spent at elevated temperatures permits the system to evolve faster, in part by more readily crossing free energy barriers.

Desmond supports replica exchange simulations in which multiple copies of the system are simulated at different temperatures, which usually range from the temperature of interest up to 700 K or more. Periodically during the simulation, attempts are made to exchange the coordinates of copies that are at different temperatures. The exchange is processed in a Monte Carlo-like process: select the systems to attempt to exchange and then use a Metropolis-like criterion to decide whether to accept the change [41]. The exchange acceptance ratio satisfies the detailed balance or balance condition so that each replica remains in equilibrium after the exchange. When many such exchanges are accepted over the course of an extended simulation, multiple systems with very different histories can visit the temperature of interest. While systems spend time at higher temperatures they explore conformational space significantly more rapidly than if they remained at the target temperature. Thus the composite trajectory at the temperature of interest may contain a more diverse collection of conformations than if multiple simulations were performed at the target temperature.

As with a regular molecular dynamics simulation each replica may be run on multiple processors. Since the simulations of each replica proceeds independently between exchange attempts the additional level of parallelization achieved by running multiple replicas is highly efficient.

Replica exchange simulations can be set up and run from the Replica Exchange panel, which you open by choosing Applications → Desmond → Replica Exchange in the main window.

In the Simulation section, you can make settings for the replica exchange job. The settings for the simulation time, recording interval, ensemble class and model system relaxation are the same as for a molecular dynamics simulation, and are described in [Section 3.4 on page 22](#). The default ensemble for replica exchange is NPT. If your highest temperature is above 373 K you might want to change the ensemble to NVT. Exchanges are done between nearest neighbors. The main specific task is to set the temperature range and temperature profile.

The temperature range is set in the Temperature range text boxes. The defaults are 300 K for the low temperature and 310 K for the high temperature. You should adjust the low and high temperature values to suitable values. There are four options for the temperature profile:

- **quadratic**—Set the temperatures by quadratic interpolation between the minimum and maximum, with the high temperature at the maximum of the quadratic curve.
- **linear**—Set the temperatures by linear interpolation between the maximum and the minimum.
- **auto**—Automatically set the number of replicas and the temperature schedule so that the acceptance ratio is approximately 30%.
- **manual**—Set the temperatures manually, by editing the temperatures in the replica table. When you select this option the table becomes editable.

Information on the temperatures is displayed in the replica table. You can edit the temperatures by selecting manual for the temperature profile. Some guidance on selecting temperatures is available in Ref. 43. Setting up the temperatures and the number of replicas for a meaningful simulation can be difficult. For assistance with this task, contact help@schrodinger.com.

The replica exchange simulation produces one trajectory for each replica, labeled `jobname_replicanum_trj`, where *num* is the index of the replica, starting from 0, and corresponds to the replica index in the replica table. You can display a temperature versus time plot of the replicas and the exchanges that were made—see [Section 5.4 on page 63](#).

3.7 Metadynamics Simulations

Metadynamics is a technique in which the potential for one or more chosen variables (“collective variables”) is modified by periodically adding a repulsive potential of Gaussian shape at the location given by particular values of the variables. These repulsive Gaussians eventually fill up the well that is being sampled, and force the calculation to sample elsewhere. For properly converged simulations, the sum of the Gaussians and the free-energy surface (FES) becomes flat, and therefore the sum of Gaussians is the negative image of the FES.

The parameters that control the accuracy of the simulation are the height and width of the Gaussian potential and the interval at which the Gaussians are added. The accuracy of the results is inversely proportional to the height for a given interval for addition of the Gaussian. However, if smaller heights are used, the simulation takes longer. The accuracy of the results increases as the time interval increases, but the simulation time also increases. The accuracy is not very sensitive to the ratio of the height to the interval, but smaller values of this ratio increase the accuracy. The width of the Gaussian should be roughly 1/4 to 1/3 of the average fluctuations of the collective variable during a free MD run.

In the Simulation section, you can make settings for the metadynamics job. The settings for the simulation time, recording interval, ensemble class and model system relaxation are the same as for a molecular dynamics simulation, and are described in [Section 3.4 on page 22](#). The default ensemble for metadynamics is NPT. The main specific task for metadynamics is defining the collective variables and the Gaussian.

The height of the Gaussian and the interval at which it is added are set in the Height and Interval text boxes, under Metadynamics parameters.

To set up a collective variable, first select an option for the type of variable: Distance, Angle, or Dihedral. (More flexible definitions can only be made from the command line.) Select Pick atoms to pick the atoms in the Workspace. The atom numbers are listed in the Atom/*N* text boxes. You can set the width of the Gaussian in the Width text box. The default values are 0.05 Å for distances, 0.03 rad (1.8°) for angles, and 0.05 rad (3°) for dihedrals.

Metadynamics

Model system

Load from Workspace Load

The system is not specified.

Simulation

Metadynamics parameters:

Height: 0.03 kcal/mol Interval: 0.09 ps

☒ Distance ☐ Angle ☐ Dihedral

☐ Pick atoms

Atom1 Atom2 Width: Wall at:

0.05

Å Å

Add Remove

Atom 1	Atom 2	Atom 3	Atom 4	Width	Wall

Simulation time (ns): total 1.2 elapsed 0.0

Recording interval (ps): energy 1.2 trajectory 4.8

Ensemble class: NPT

Temperature (K): 300.0 Pressure (bar): 1.01325

Surface tension (bar·Å): 4000.0

☒ Relax model system before simulation

Relaxation protocol: Browse...

Advanced Options...

Desmond
Developed by D. E. Shaw Research

Start... Write... Reset Close Help

Figure 3.5. The Metadynamics panel.

For distance variables, you can place a barrier (“wall”) at a given distance in the Wall at text box. Introducing a wall prevents the system from moving too far in the direction defined by the collective variable.

Once you have set up the variable, you can add it by clicking Add. The variable is added to the table, and is used in the simulation. To remove a variable from the simulation, select it in the table and click Remove. Setting up more than two collective variables can create sampling problems.

More flexibility in the collective variables is available from the command line, via `multisim`. You can use ASL expressions to define a group of atoms whose center of mass is used as the variable, for example.

3.8 Simulations on Systems with Membranes

Simulations of systems that contain membranes require some special consideration. This is because nearly all current all-atom membrane potential models in existence do not, on their own, maintain the appropriate surface areas on the time scale of tens of ns in simulations of pure membranes. If non-lipid components make up a significant fraction of the membrane region (such as a protein in a relatively small amount of lipid), this issue is much less pronounced and many not require special treatment. In this case the semi-isotropic NPT ensemble may work well. However, if the simulated membrane is pure or only contains a small solute (e.g. ligand-sized) the following practical approach may be useful.

When a solute is placed in a pure membrane, some lipids are usually removed to make room for the new molecule during the system building process. As a result, adjustment of the surface area of the solute + membrane system is often needed. This can usually be done using a fairly short semi-isotropic simulation of up to about 0.5 ns. When simulating beyond that time range it is recommended to switch to either a constant surface area, constant normal pressure simulation (NPAT) or a constant surface tension simulation (NP γ T). If the latter is selected we suggest using a surface tension of 2000 bar/Å for DPPC and 4000 bar/Å for POPE and POPC. We recommend examining the results of all membrane simulations carefully.

It can be difficult to relax freshly built protein-membrane systems. In particular, penetration of the water between the protein than the lipids can be problematic and require very lengthy simulations to correct. A relaxation protocol that should reduce or eliminate such problems is available by running the script `relax_membrane.py` from the command line.

To use the membrane relaxation protocol:

1. Save your newly built protein-membrane system in a CMS file (referred to here as *protein-membrane.cms*).
2. Run the script to prepare the necessary input files:

```
$SCHRODINGER/run relax_membrane.py -i protein-membrane.cms  
-t temperature -j protein-membrane
```

3. Run the membrane relaxation protocol using the command

```
$SCHRODINGER/utilities/multisim -JOBNAME protein-membrane  
-HOST myhost -mode umbrella -cpu cpus -i protein-membrane-in.cms  
-m protein-membrane.msg -o protein-membrane-out.cms
```


This process may take hours to days since it equilibrates the system in stages for about 1.2 ns. The file *protein-membrane-out.cms* should be reasonably well equilibrated and can be used as input for the production simulation for your study.

3.9 Setting Options for Desmond Simulations

The default settings used in the Desmond panels were selected to produce good results in the majority of cases. At times, you may want greater control over the parameters of the calculation. The Advanced Options dialog box provides access to advanced options for control of the simulation or the minimization, such as the frequency of data output, integration time step sizes, thermostat and barostat parameters, restraints, cutoff radii, and so on.

To open the Advanced Options dialog box, click Advanced Options in the Desmond panel that you have open. This panel has several tabs, which are described in the following subsections.

- Integration tab
- Ensemble tab
- Minimization tab
- Interaction tab
- Restraints tab
- Output tab
- Misc tab

The selection of tabs that is displayed depends on the task. For minimizations, only the Minimization, Interaction, Restraints, and Misc tabs are displayed. For MD simulations all but the Minimization tab are displayed.

The settings in this dialog box and the settings in the Desmond panels are not entirely independent, and can affect each other. For example, changing the far time step can affect the values of the recording periods in the panel, because the latter are automatically rounded by the far time step. As another example, changing the temperature or pressure in the Desmond panel updates the temperature or pressure parameters in the dialog box. Changes in the Desmond panel take effect immediately and update parameters in the dialog box, whereas changes made in the dialog box only take effect when you click OK or Apply.

If you want to clear changes that have not been committed with the Apply button, click Reset. Any changes made since the last set of changes were committed are discarded, and the values in the dialog box are reset to the last set committed.

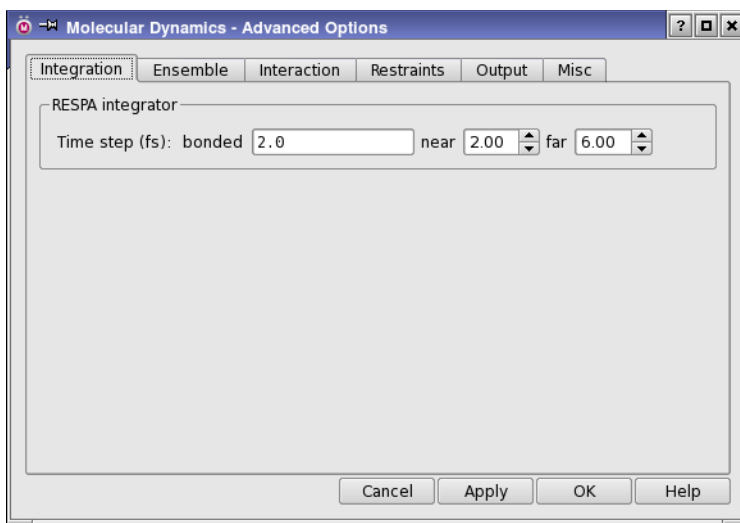


Figure 3.6. The Integration tab of the Advanced Options dialog box.

3.9.1 The Integration Tab

In this tab, you can set parameters for the integration algorithm. The tab has a single section, RESPA integrator.

RESPA integrator section

Specify the time steps in fs for bonded, near, and far, by entering values in the text boxes, or using the arrow buttons to change the value in increments of the bonded time step. Because the bonded, near, and far time steps must maintain a certain ratio, when a new value is set for the bonded time step, the other two time steps are automatically updated according the current ratio. Changing the near or far time steps adjusts this ratio.

Selecting Set time step automatically based on constraint setting couples the RESPA time step settings with those in the Constraint section. The time steps will be automatically set based on the settings in the Constraint section, and are not available for editing.

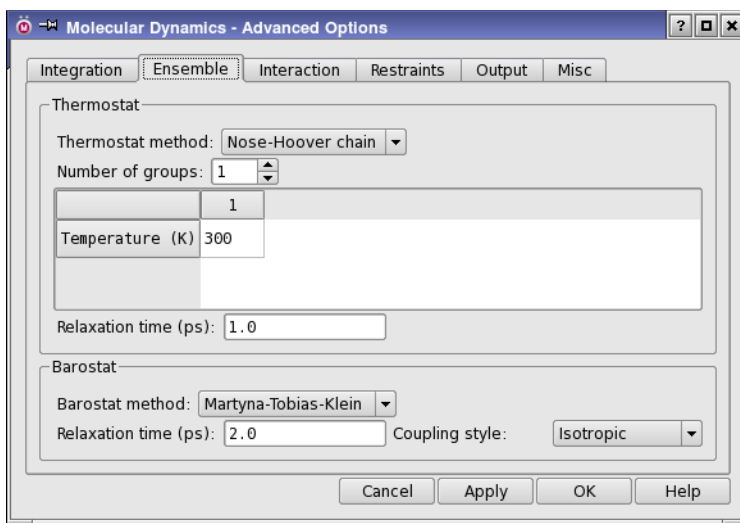


Figure 3.7. The *Ensemble* tab of the *Advanced Options* dialog box.

3.9.2 The Ensemble Tab

In this tab you can set the thermostat method and the barostat method and adjust the settings for these methods. The thermostat method and the barostat method are coupled: the choice you make from the Thermostat method option menu changes the selection from the Barostat method option menu, and vice versa. The exception is that you can choose None for the barostat method for any of the thermostat methods.

The Thermostat method option menu offers four choices: Nose-Hoover chain, Berendsen, Langevin, and None.

Although in most circumstances, only one thermostat group is needed, you can specify multiple thermostat groups by entering the number of groups in the Number of groups text box and supplying information on these groups in the Thermostat group settings table. The maximum number of groups is 8. The selection of atoms that is in each group can be set up in the Misc tab, by defining multiple groups named `thermostat`, with the group numbers corresponding to the entries in the Values column. Any atoms not explicitly added to a group are automatically assigned to group 0, the default group. This means that you do not need to define a group if you only want to use one thermostat, and that you only need to define groups for the extra thermostats, starting from thermostat 1.

The Thermostat group settings table provides text boxes for making settings for each thermostat group. The settings that can be made are:

- Temperature (K)
- Relaxation time (ps)

The Barostat method option menu also offers four choices: Martyna-Tobias-Klein, Berendsen, Langevin, and None. For each of these methods you can set the relaxation time (ps) in the Relaxation time text box and choose a coupling style from the Coupling style option menu. The coupling style choices are Isotropic, Semi-isotropic, Anisotropic, and Constant area. The pressure used is atmospheric pressure.

3.9.3 The Minimization Tab

In this tab you can set parameters for the minimization, and also specify the output file. Minimization is performed with the LBFGS method, with an optional steepest descent initial phase.

The Minimizer section provides the following controls:

- LBFGS vectors—Specify the number of history vectors used by the LBFGS minimizer for the update of the Hessian. The maximum is 6.
- Minimum SD steps—Specify the minimum number of steepest descent steps to be used before switching to the LBFGS minimizer.

In the Output section you can specify the name of the structure output file. You can use \$JOBNAME as a variable representing the job name that you will set when you start the job or write out the input files.

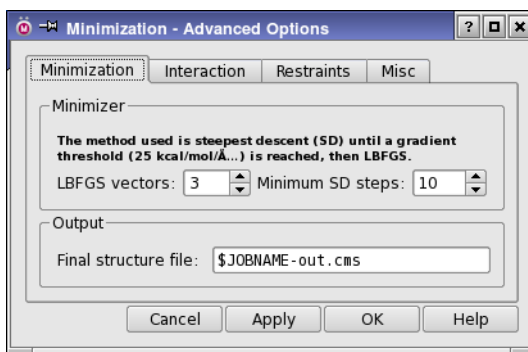


Figure 3.8. The Minimization tab of the Advanced Options dialog box.

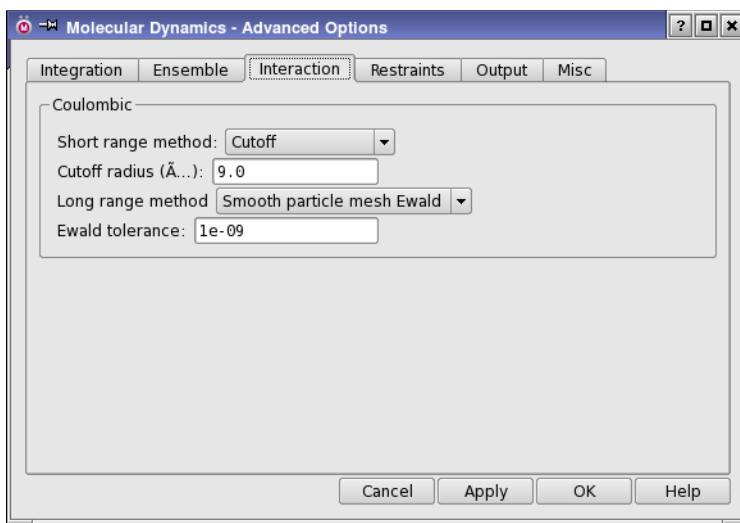


Figure 3.9. The Interaction tab of the Advanced Options dialog box.

3.9.4 The Interaction Tab

In this tab you can specify how the short-range and long-range Coulombic interactions are handled.

To define the short-range region, choose a method from the Short range method option menu. The controls below this menu depend on the method chosen, which can be one of the following:

- Cutoff—Enter a value in the Cutoff radius text box. The default is 9.0 Å.
- Potential tapering—Specify the range in angstroms over which the potential is tapered off in the two Tapering range text boxes.

There are two choices for handling the long-range Coulombic interactions:

- Smooth particle mesh Ewald—use the smooth particle mesh Ewald method. This method requires a tolerance to be set in the Ewald tolerance text box. This tolerance affects the accuracy of the long-range Coulombic interactions. The smaller the tolerance is, the more accurate the computation of the long-range Coulombic interactions is, but the simulation will be correspondingly slower.
- None—use the unmodified Coulomb interaction.

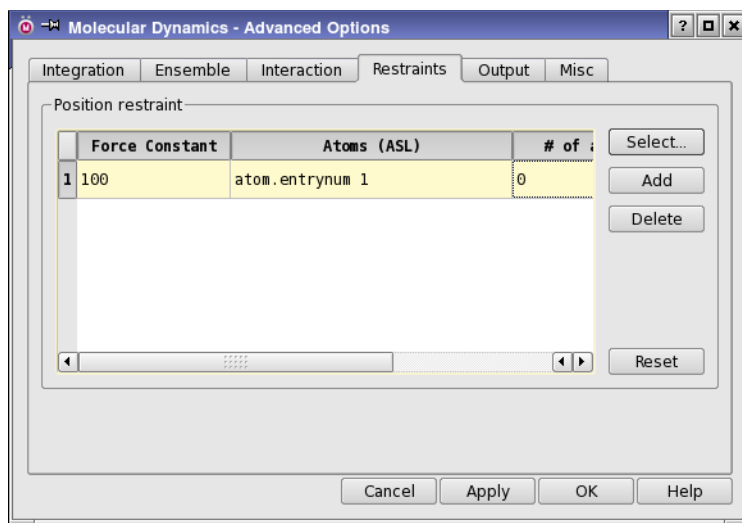


Figure 3.10. The *Restrains* tab of the *Advanced Options* dialog box.

3.9.5 The Restrains Tab

In this tab you can specify restraints on atom positions. A restraint is defined by a set of atoms and a force constant. The restraints are listed in the restraints table. The Atoms column is filled in automatically when you click **Select** and use the **Atom Selection** dialog box to specify the atoms. You must enter the force constant in the table manually, by editing the table cell.

To manage the restraints, you can use the buttons beside the table:

- **Select**—Opens the **Atom Selection** dialog box to specify the atoms for the selected restraint. Only available if a single row is selected in the table.
- **Add**—Adds a row to the restraints table so that a new restraint can be defined.
- **Delete**—Deletes the selected restraints.
- **Reset**—Resets the table to its default state.

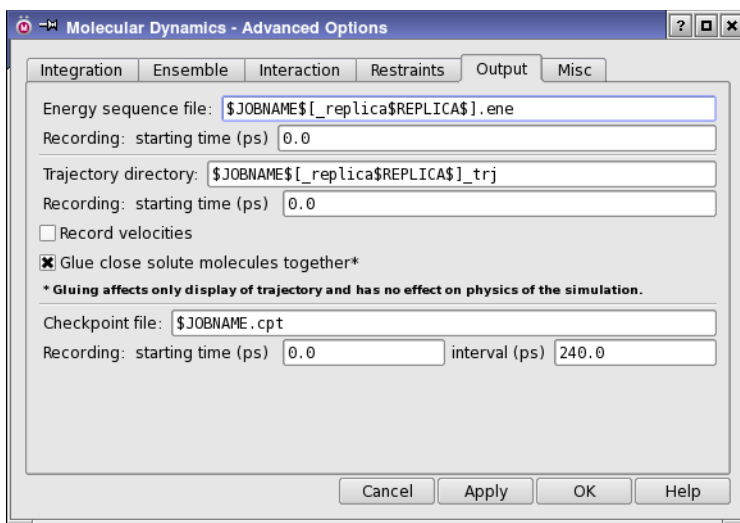


Figure 3.11. The Output tab of the Advanced Options dialog box.

3.9.6 The Output Tab

In this tab you can set names for various output files, and set the times for which recording in these files begins, and the update frequency. For each file there is a starting time text box, in which you can enter the starting time for recording of information to this file or adjust the starting time in increments of 50 times the far time step (except for the checkpoint file, where the increment is 5000 times the far time step). Some files also have an interval text box, in which you can enter or adjust the recording interval (in the same increments as the starting time). The files are listed below, with any extra controls. The intervals for the energy sequence file and for the trajectory can be set in the Desmond panel.

- **Energy sequence file**—This file contains a sequence of various energies of the system.
- **Trajectory directory**—This directory is used by Desmond to periodically write out files that record coordinates and velocities (optional) of all particles in the system at a particular point in the simulation. You can provide a title for the trajectory, and you can select Record velocities if you want the velocities to be recorded along with the coordinates.

To ensure that associated solutes appear together in the trajectory rather than on opposite sides of the simulation box, you can select Glue close solute molecules together. This option only affects the way the trajectory is displayed.

- **Checkpoint file**—This file contains information that can be used to restart an interrupted simulation. Checkpoint files permit bitwise continuation of simulations, so they can be large, and should be saved infrequently if at all.

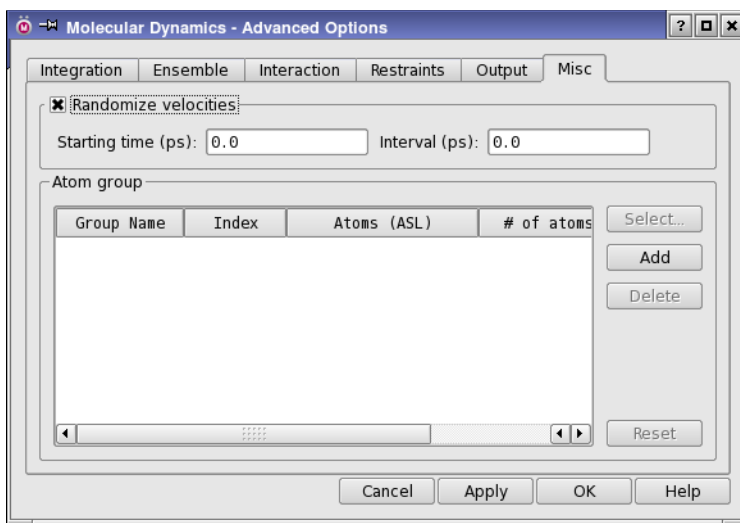


Figure 3.12. The Misc tab of the Advanced Options dialog box.

3.9.7 The Misc Tab

This tab provides access to various options that do not fit into the other categories. In this tab you can specify when the velocities are randomized, and specify atom groups for special treatment in the simulation.

You can set the starting time and the interval at which velocities are randomized in the Randomize velocities section. By default velocities are randomized at the beginning of a calculation, and not randomized again. For some kinds of simulations (e.g. in the gas phase), periodic randomization of velocities can improve the sampling.

In the Atom group section you can specify atom groups. Atom groups are used for various special treatments of atoms in the simulation.

You can define multiple groups with the same name but a different index by setting the index in the Value column. If you use multiple thermostats, for example, you can define the atoms in each thermostat group by naming the group thermostat and setting the index to the thermostat group number in the Ensemble tab. Group 0 is the default group, to which all unassigned atoms automatically belong.

The atom groups are listed in the table. The Atoms column is filled in automatically when you click Select and use the Atom Selection dialog box to specify the atoms. Otherwise you can edit this column to specify the ASL expression for the atoms in the group. The number of atoms defined by the ASL expression is shown in the No. of Atoms column.

Beside the table are the following buttons:

- **Select**—Opens the Atom Selection dialog box to specify the atoms for the selected group. Only available if a single row is selected in the table.
- **Add**—Adds an atom group. Clicking this button adds a row to the table.
- **Delete**—Deletes the selected groups. This operation can only be done on user-defined groups.
- **Reset**—Resets the table to its default state.

3.10 Running a Simulation Job

Once you have finished making settings, you can click **Start** to set up the job parameters and run the job, or you can click **Write** to write out the input files and run the job from the command line. For information on running from the command line, see [Chapter 6](#).

When you click **Start**, the Start dialog box opens. The common features of this dialog box, such as the Output section, the Job name, and CPUs text boxes, and the Host option menu, are described in [Section 2.2](#) of the *Job Control Guide*. These features allow you to direct the job to the appropriate host with the desired number of processors, and decide how to incorporate the output into the Maestro project.

For Desmond jobs, the choice of the number of processors has some special requirements. Desmond uses a Cartesian domain decomposition of the simulation for efficient processing. This decomposition is described in the *Desmond User's Guide*, provided by D. E. Shaw Research. The details of the decomposition can be displayed in the Start dialog box by clicking the Details button. You can control the decomposition by entering values in the x, y, and z text boxes. The values you enter must be a power of 2, 3, or 5, or products of these powers. The decomposition is done automatically if you enter a number in the CPUs text box. However, if you enter a number that is not a suitable product of powers, the actual number of processors used is the largest product of powers that is smaller than the number you entered. The number of processors actually used is displayed in the details section. To hide the details, click the Details button again.

Before you can run Desmond jobs in parallel, you must configure the hosts file and any queues that you want to use. Details are given in [Chapter 7](#) of the *Installation Guide*.

When the job is submitted to a queueing system, the processors requested are all allocated to the master job, and the subjobs are run directly on these processors, without being submitted to the queueing system. (This is called “umbrella” mode—see [Section A.1.2 on page 106](#).)

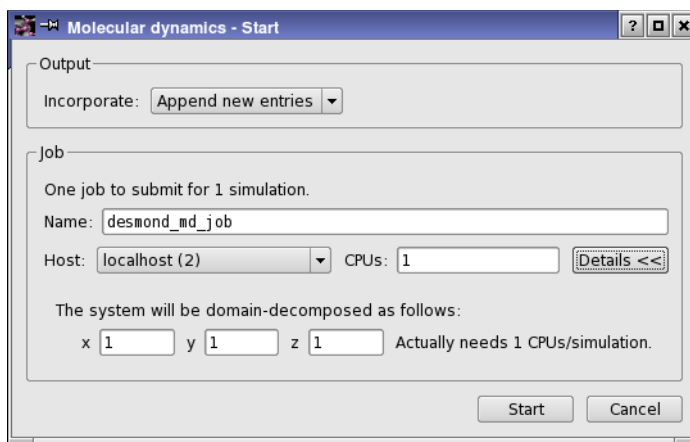


Figure 3.13. The Start dialog box.

Running FEP Simulations

Maestro provides panels for quickly setting up FEP simulations for certain common scenarios:

- Mutation of ligand functional groups: a group attached to a ligand by a single bond is mutated to some other group.
- Aromatic ring atom substitution: an atom (with its attached hydrogens) is mutated into another atom, usually a heteroatom.
- Protein residue mutation: the side chain of a protein residue is mutated into that of another residue.

In these three scenarios, the relative binding free energy of the ligand can be calculated. For the first two, the solvation free energy of the ligand relative to the original ligand can be calculated.

Total (absolute) solvation free energies can also be calculated for a ligand or a molecule, which is done by annihilating the chosen molecule in the FEP simulation.

Free energy perturbation calculations are resource-intensive. While it is possible with the FEP panels to set up multiple mutations in a single calculation, we recommend that only one mutation be performed per calculation due to the resource requirements.

4.1 FEP Panels

The FEP panels are designed to make setting up FEP jobs as easy as possible: most of the computational details, such as setting the force field parameters, solvation, relaxation of the system, and simulation time, are taken care of automatically. These predetermined parameters for the FEP calculations should work for most systems, but you can also write out the input files and then modify the simulation parameters in the FEP panel—see [Section 4.8 on page 53](#). (Note that checkpoint files are not written out for the relaxation stages.)

The panels for these calculation types have a common design. Each panel has a Define Perturbation tab, in which you set up the systems to be simulated, and a Plan Calculation tab, in which you choose the environment (complex, pure solvent, vacuum) for the ligand and choose the FEP protocol, which includes the ensemble. Information on setting up the system is contained in the next few sections, followed by a section describing the Plan Calculation tab.

The panels have action buttons at the bottom:

- **Start**—Opens the Start dialog box, in which you can set the job parameters such as job name, host, number of CPUs, and so on, and then start the job. The general layout of this dialog box is described in [Section 2.2](#) of the *Job Control Guide*. The Desmond-specific features of the Start dialog box and the issues in choosing the number of CPUs are described in [Section 3.10 on page 39](#).
- **Write**—Writes the input files for the job but does not start the job. Opens a dialog box in which you can specify a job name, which is used to name the files. This capability is useful if you want to change the FEP protocol.
- **Reset**—Resets the entire panel to its initial state. This includes clearing the Workspace. If you want to use the same structures, you will have to redisplay them before proceeding.

4.2 Ligand Functional Group Mutation

In ligand mutation, a group on the ligand molecule, called the *substitution group*, is replaced with selected fragments. These fragments are chosen from a list of small, predefined fragments. The fragments can be adjusted to reduce steric clashes, and can be modified by using the tools on the Build toolbar or in the Build panel.

The FEP calculations for ligand mutation are set up in the Ligand Functional Group Mutation by FEP panel, which you open by choosing one of the following:

- Applications → Desmond → Ligand Functional Group Mutation by FEP
- Tasks → Binding Energy Estimation → Ligand Functional Group Mutation → Molecular Dynamics FEP

To run a ligand mutation job:

1. Include in the Maestro Workspace the system that contains the ligand to be mutated, the receptor protein (optional), and any other key molecules of the system.

This step can be performed prior to opening the panel. You can use an existing model system that may, for example, include a membrane. If you do, note that the SPC water model is used in the FEP simulations, regardless of the choice made for your model system.

2. Select **Pick the attachment bond**, then pick the attachment bond in the Workspace, at the end toward the substitution group.

This bond is the bond at which the substitution will take place. When the bond is picked, a green arrow is drawn over the bond, pointing to the substitution group, and the ligand core is colored with green carbons. If you clicked on the wrong end of the bond, click again to correct the error. Once focus is returned to the panel, this option is deselected.

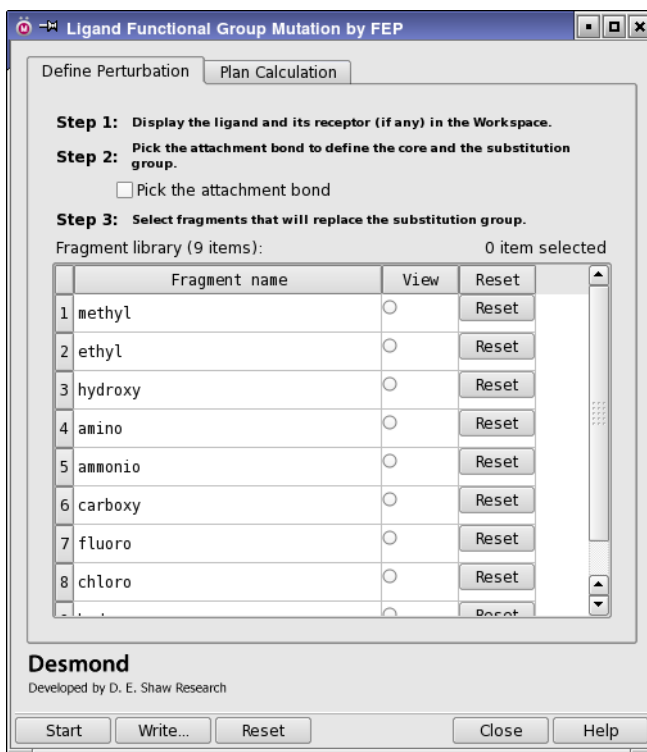


Figure 4.1. The Ligand Functional Group Mutation by FEP panel, Define Perturbation tab.

3. In the Fragment library table, select the fragments that you want to use to replace the original substitution group, by selecting their table rows.

Use shift-click and control-click to select multiple table rows. You can view the mutated ligand by clicking in the View column.

Since each FEP mutation requires a large amount of CPU time and space, we recommend performing only one mutation in a given calculation.

4. (Optional) Display the mutated ligand and make adjustments to the fragment.

The adjusted fragment is used as the starting point in the FEP simulations. It is possible that the fragment may have clashes with the protein in its default orientation or be in an inappropriate conformation. The barriers between local minima in the potential energy surface can be large, and thus take a long time to cross during a simulation. Adjusting the conformation can reduce clashes and help the simulation to sample the appropriate conformations.

Once the fragment is displayed, you can reorient the fragment with the adjustment tools, which are available from the Adjust toolbar button:



or with the local transformation tools, which are available from the Transform toolbar button:



For more information on these tools, see [Section 5.9](#) of the *Maestro User Manual*.

You can also add to the fragment using the Build toolbar or the Build panel—for example, replacing a hydrogen with a methyl or a hydroxyl group. Since the quality of the results goes down as you increase the size of the fragment, it is not advisable to add too much to the fragment.

You can revert to the original fragment by clicking the button in the Reset column. You should not make modifications to the ligand core or other molecules in the system.

5. In the Plan Calculation tab, select the calculation types:
 - For binding free energy calculations, select In complex and In pure solvent.
 - For solvation free energy calculations, select In pure solvent and In vacuum.
6. Choose the FEP protocol you want to use for each calculation type from the FEP protocol option menu.

You should choose the same ensemble type for both the complex and pure solvent calculations.

7. Click Start, set the job parameters in the Start dialog box, and click Start in the dialog box to run the job.

4.3 Ring Atom Mutation

Ring atom mutation changes a selected ring atom to a different atom type while preserving the hybridization of the atom. To open the Ring Atom Mutation by FEP panel, choose one of the following:

- Applications → Desmond → Ring Atom Mutation by FEP
- Tasks → Binding Energy Estimation → Ligand Ring Atom Mutation → Molecular Dynamics FEP



Figure 4.2. The Ring Atom Mutation by FEP panel, Define Perturbation tab.

To run a ring atom mutation job:

1. Include in the Maestro Workspace the system that contains the ligand to be mutated, the receptor protein (optional), and any other key molecules of the system.

This step can be performed prior to opening the panel. You can use an existing model system that may, for example, include a membrane. If you do, note that the SPC water model is used in the FEP simulations, regardless of the choice made for your model system.

2. Select Pick the ring atom, then pick the ligand atom to be mutated in the Workspace.

When the atom is picked, it is rendered in ball-and-stick, and the ligand is colored with green carbons. The Mutation targets table is displayed. Once focus is returned to the panel, this option is deselected.

3. In the Mutation targets table, select the targets that you want to use to replace the original ring atom by selecting their table rows.

Use shift-click and control-click to select multiple table rows. You can view the mutated ligand by clicking in the View column. If you want to add a functional group as well as mutate the ring atom, you can use the build tools to add to the mutated structure.

4. In the Plan Calculation tab, select the calculation types:
 - For binding free energy calculations, select In complex and In pure solvent.
 - For solvation free energy calculations, select In pure solvent and In vacuum.
5. Choose the FEP protocol you want to use for each calculation type from the FEP protocol option menu.
6. Click Start, set the job parameters in the Start dialog box, and click Start in the dialog box to run the job.

4.4 Protein Residue Mutation

In protein residue mutation, a protein residue is mutated into another residue in the presence of a ligand, and the change in the ligand binding free energy due to the mutation is calculated.

The FEP calculations for protein residue mutation are set up in the Protein Residue Mutation by FEP panel, which you open by choosing one of the following:

- Applications → Desmond → Protein Residue Mutation by FEP
- Tasks → Binding Energy Estimation → Protein Residue Mutation

To run a protein residue mutation job:

1. Include in the Maestro Workspace the system that contains the ligand to be mutated, the receptor protein, and any other key molecules of the system.

This step can be performed prior to opening the panel.

2. Select Pick atom to identify ligand, then pick a ligand atom in the Workspace.
3. In the Define Mutations section, select Pick residue.
4. Pick a protein residue in the Workspace.

The residue is added to the table as a mutation site.

5. In the Substitutes column, double-click in the table cell and choose the residues from the list that you want to use to replace the original.
6. Repeat [Step 4](#) and [Step 5](#) for as many mutations as you want to perform.

You can pick the same residue multiple times to mutate it to different residues. If you pick different residues, each residue that you pick is mutated.

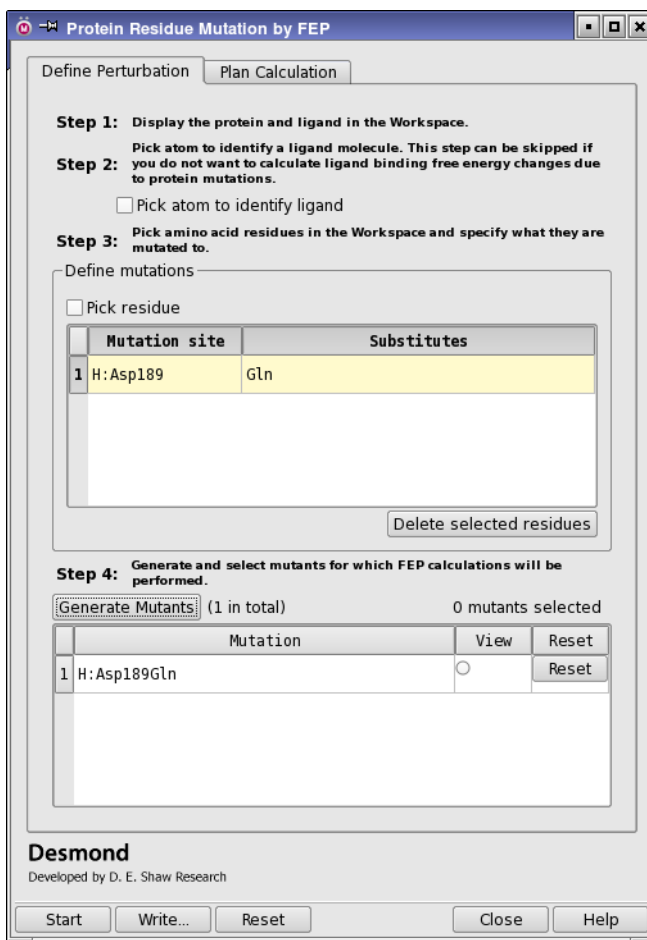


Figure 4.3. The Protein Residue Mutation by FEP panel, Define Perturbation tab.

7. Click Generate Mutants.
8. The table is filled with the mutations that you have defined. For each of the original residues that you picked, all possible combinations of mutations are added to the table. For example, if you picked two different residues, and for each you specified three substitutes, the table would list nine mutations, corresponding to each possible combination.
9. In the mutants table, select the mutations that you want to use, by selecting their table rows.

Use shift-click and control-click to select multiple table rows. You can view the mutated protein by clicking in the View column.

Once the residue is displayed, you can reorient the side chain with the adjustment tools, which are available from the Adjust toolbar button:



or with the local transformation tools, which are available from the Transform toolbar button:



For more information on these tools, see [Section 5.9](#) of the *Maestro User Manual*.

You can also modify the residue using the Build toolbar or the Build panel—for example to replace sulfur with selenium, or to phosphorylate the residue. Since the quality of the results goes down as you increase the size of the perturbation, it is not advisable to add too much to the residue.

You can revert to the original residue orientation and structure by clicking the button in the Reset column. You should not make modifications to the ligand core or other molecules in the system.

10. In the Plan Calculation tab, choose the FEP protocol you want to use for each calculation type from the FEP protocol option menu.

You should choose the same ensemble type for both the complex and pure solvent calculations.

11. Click Start, set the job parameters in the Start dialog box, and click Start in the dialog box to run the job.

For calculations that are not defined by a simple protein, ligand, and water model (such as membrane proteins or calculations in salt), you can use existing model systems for input. To do this you should prepare two model systems, one with and one without the ligand. You should run the In complex calculation only, for both systems, to ensure that the model system is used as is for both perturbations. The relative binding free energy can be calculated from the difference between the two free-energy differences.

4.5 Total Solvation Free Energy Calculation

Calculating the absolute solvation free energy for a molecule involves running an “annihilation” FEP simulation, in which the selected molecule is removed from the system. See [Reference 45](#) for a study of absolute solvation free energies.

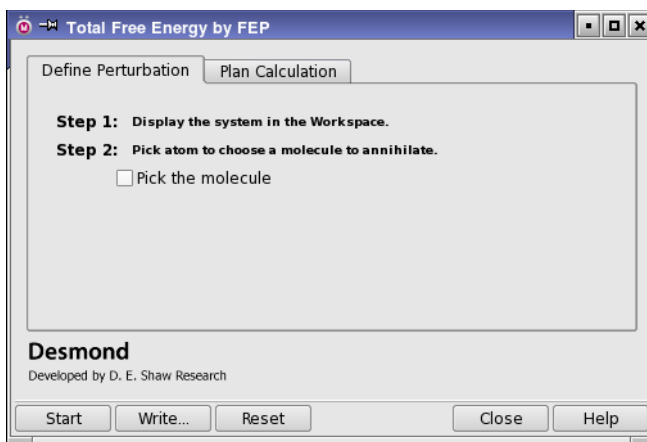


Figure 4.4. The Total Free Energy by FEP panel, Define Perturbation tab.

To run an annihilation job:

1. Include in the Maestro Workspace the system that contains the molecule to be annihilated.

This step can be performed prior to opening the panel.

2. Select Pick the molecule, then pick the molecule in the Workspace.

When the molecule is picked, it is colored with green carbons. Once focus is returned to the panel, this option is deselected.

3. In the Plan Calculation tab, ensure that In pure solvent is selected.
4. Choose the FEP protocol you want to use for each calculation type from the FEP protocol option menu and make any related settings.

The default simulation time is 2 ns, which is recommended for accurate results.

5. Click Start, set the job parameters in the Start dialog box, and click Start in the dialog box to run the job.

4.6 Selecting the Environment and FEP Protocol

The selection of the ligand environment and FEP protocol is made in the Plan Calculation tab. This section explains the choices, which were referred to in the preceding sections.

The FEP simulations can be performed in three environments: in the protein-ligand complex (with solvent), in pure solvent, and in vacuum. This permits the calculation of binding free energies, by selecting In complex and In pure solvent, and of solvation free energies, by

selecting In pure solvent and In vacuum. Selecting just one of the environments does not on its own produce a meaningful result.

The FEP protocol defines in detail how system is solvated, relaxed, and simulated. Four main protocols are provided in the FEP protocol option menu for the complex and pure solvent: two ensembles, Desmond NPT and Desmond NVT, with two options for the relaxation part (standard, or quick relaxation). The protocol for the simulation is the same for both ensembles (NVT and NPT). For calculations in vacuum, there is the choice of the standard or quick relaxation. You can set the temperature for all the simulations in the Temperature text box.

In addition you can read in a protocol by choosing User defined from the FEP protocol option menu. A file name text box and Browse button is displayed. You can enter the name of the protocol file (which has a .msj extension) in the text box, or navigate to it in the file selector that opens when you click Browse.

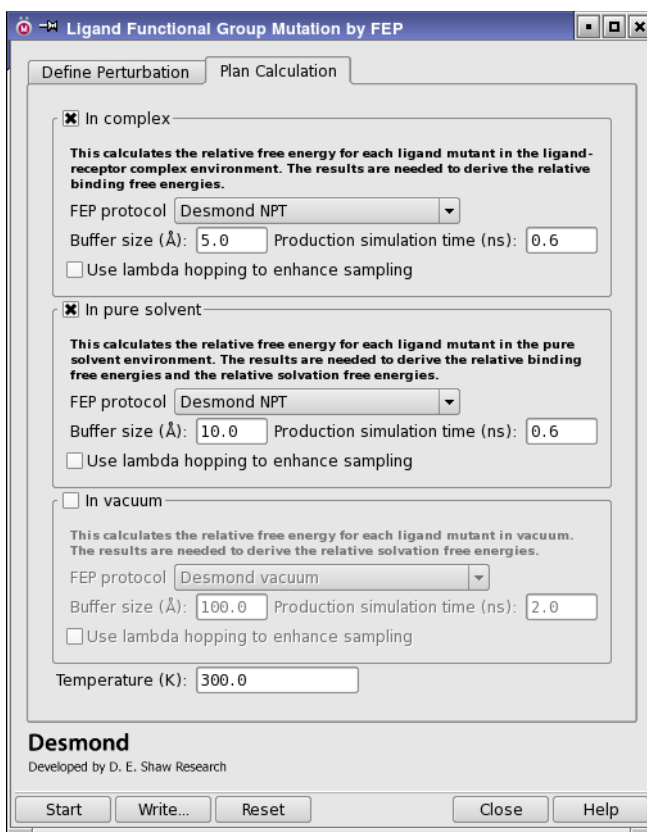


Figure 4.5. The Plan Calculation tab.

The default protocol is as follows. The original system is solvated by adding SPC water molecules with a buffer distance of 5 Å for complexes and 10 Å for pure solvent. The vacuum simulation uses a buffer distance of 100 Å. The system goes through a relaxation process that includes two minimizations followed by 4 short molecular dynamics simulations. The production simulation is run for 0.6 ns for each lambda window, and 12 windows are used for each perturbation. You can change the buffer size and the production simulation time for each environment in the Buffer size and Production simulation time text boxes. If you are using an existing model system, the buffer size is ignored for the complex, but is used for the pure solvent.

Lambda hopping is a form of Hamiltonian replica exchange in which exchanges of coordinates for systems simulated in adjacent FEP windows are attempted. For mutations involving conformational flexibility, lambda hopping can improve performance. To use this method, select Use lambda hopping to enhance sampling.

After the production simulations, the results are collected and analyzed using the Bennett method. The final result (free energy) for each perturbation is recorded as an entry-level property in the final Maestro output file.

4.7 FEP Results

The results of FEP calculations usually need some further processing to produce the quantities of interest. The most common of these are discussed in the sections below.

4.7.1 Relative Free-Energy Differences

The estimated changes in free energy for FEP calculations are recorded as structure-level properties in the mutated structures (rather than in the original structures) in the output structure file. For ligand functional group mutation, protein residue mutation and ring atom mutation, these values are displayed in Maestro as *dG jobname*, and recorded in the file as *s_des_dG_jobname*, where *jobname* is the name for the multisim job and usually ends in *_vacuum*, *_solvent* or *_complex* for the corresponding environments (vacuum, pure solvent, and complex). These values are strings rather than real numbers because they include the estimated uncertainty. To obtain a relative solvation free energy, subtract the vacuum result from the solvent result. Similarly, relative binding free energies are calculated by subtracting the solvent result from the complex result.

You can use the script `calculate_ddg.py` to take the difference between the values in two different environments to provide a $\Delta\Delta G$ value. To run this script, use the following command:

```
$SCHRODINGER/run -FROM desmond calculate_ddg.py jobname1-out.mae  
jobname2-out.mae [output-file.mae]
```

where *jobname1-out.mae* and *jobname2-out.mae* are the Maestro output structure files from the two FEP calculations. The output file *output-file.mae* is optional; the default name is *ddG.mae*. This file is in Maestro format and contains the ΔG value from both input files. The final $\Delta\Delta G$ value is recorded for each mutant as an entry property *s_des_ddG_jobname1-jobname2*, which is displayed in the Project Table as *ddG_jobname1-jobname2*. The atom coordinates in the output file are copied from the first input structure file.

4.7.2 Total Solvation Free Energies

The total absolute free energies calculated by Desmond are recorded in the output structure file as structure-level properties for the solute structures. The property names are *s_des_dg_jobname_transfer* and *s_des_dg_jobname_solvation*. The former, the transfer free energy, uses the same concentration in both the vacuum and solution phases while the latter, the solvation free energy, uses standard state concentrations in the two phases (1 bar in vacuum and 1 Molal or 1 mole of solute per kg of solvent in solution).

When charged molecules are deleted or created in absolute free energy calculations, finite size effects can be significant, particularly for the pure solvent FEP calculations. A script, *calculate_correction.py*, has been provided to provide a correction to the free energy once the FEP calculation is complete. To run this script, use the following command:

```
$SCHRODINGER/run -FROM desmond calculate_correction.py jobname [data-dir]
```

where *jobname* is the name of the $\lambda=0$ simulation job for the original FEP calculation (e.g., *myfepjob_complex_4_lambda0*). This program assumes the input and output file names are derived from *jobname*. Specifically, this program looks for the following files and directories: *jobname-in.cms*, *jobname-in.cfg*, and *jobname_trj*. *jobname-in.cms* and *jobname-in.cfg* must be the input files of the simulation, and *jobname_trj* is the trajectory directory produced during the simulation. The optional argument *data_dir* can be used to designate the name for the directory in which to look for the data files. If this argument is missing, the default directory name is *jobname*.

This program reviews the trajectory and calculates the correction. The final correction is printed to standard output. During the process, the program writes out several files: *jobname_correct1.cfg*, *jobname_correct1.log*, and *jobname_correct1.dat*. Depending on your system, it may also write out the analogous files: *jobname_correct2.cfg*, *jobname_correct2.log*, and *jobname_correct2.dat*. You need not be concerned with the content of these files.

4.7.3 FEP Analysis

Judging the accuracy and convergence of free energy calculations is difficult, particularly if no additional data is available. Desmond FEP jobs generate additional information for relative free energy and total solvation free energy calculations, which can help to identify accuracy and convergence problems. This information includes plots of the distributions of the perturbation energies for mutating between adjacent windows, the time evolution of the overall free energy and the corresponding estimates for the individual perturbations. The information is available in the tar file of the `fep_analysis` stage for the calculation for absolute solvation free energies and each of the two FEP calculations needed to generate a $\Delta\Delta G$ value for relative free energy calculations. These tar files are usually the highest numbered ones.

For instance, for ligand functional group mutations in the presence of a protein these files would normally be called `jobname_solvent_12.tgz` and `jobname_complex_12.tgz`. These files can be extracted using the command:

```
tar -xzf tarfilename
```

Inside the extracted directory there is an HTML file, which you can open in a web browser to view the various plots.

4.8 Customizing and Restarting FEP Simulations

FEP jobs perform a series of MD simulations for the calculation of the free energy difference between two systems. These simulations are set up in the specialized FEP panels described earlier in this chapter. The FEP panel allows you to adjust FEP protocols set up in these panels, and to rerun lambda windows for FEP jobs that failed. To open the FEP panel, choose Applications → Desmond → FEP in the main window.

The settings for the simulation time, recording interval, ensemble class, temperature, pressure, surface tension, and model system relaxation are the same as for a molecular dynamics simulation, and are described in [Section 3.4 on page 22](#).

There are two options for the type of FEP that can be performed, that correspond to the types of FEP that are set up with the specialized FEP panels:

- **Mutation**—Mutate one set of atoms into another. The atom set may be a molecule or a fragment. Select this option for ligand functional group mutation or ring atom mutation.
- **Total free energy**—Select this option for total free energy simulations, in which a set of atoms is annihilated by the perturbation.

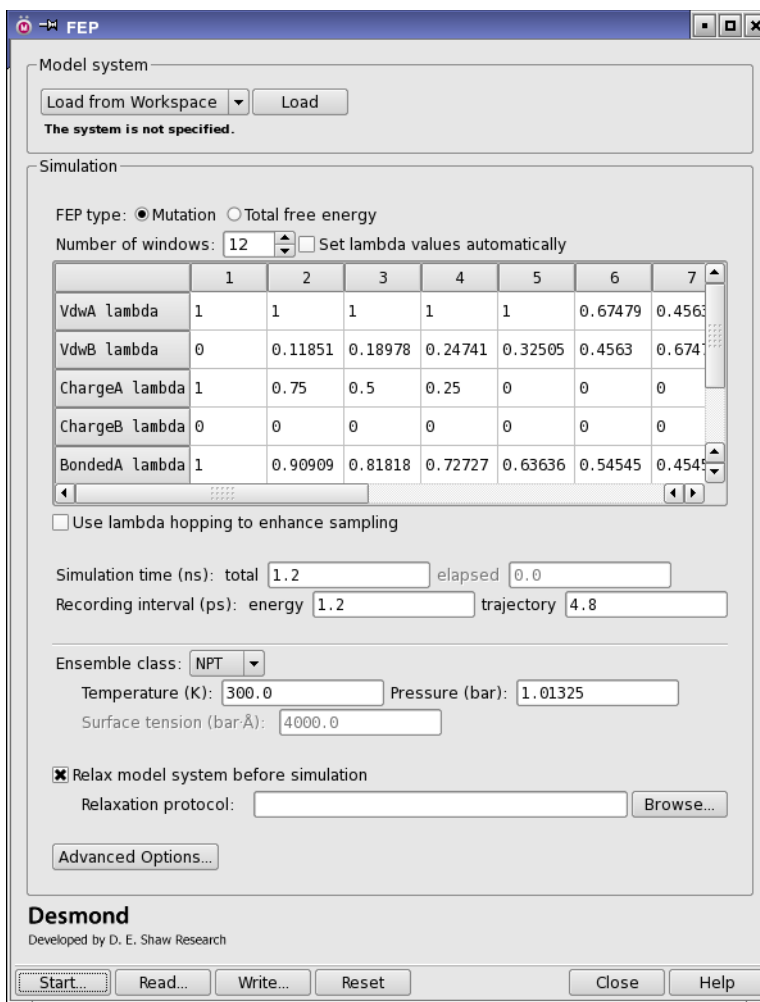


Figure 4.6. The FEP panel.

The number of lambda windows you want to use can be entered in the Number of windows text box. Each window is listed in the table below, with values for each of the six lambda parameters, Van der Waals, Charge and Bonding, for the two systems, labeled A and B.

You can set the lambda values automatically, which is the default, or you can deselect Set lambda values automatically and edit the values for each window and each type of lambda. The lambda values must be within the range [0, 1]. The background color of the cell is changed to pink if the value is out of range, to warn that it must be set within the range.

The windows can be selected or deselected for the simulation using the buttons in the Window index row. For a new FEP job, all windows should normally be selected for simulation. If a simulation for a particular lambda window needs to be re-run, select only that window.

For mutation FEP, you can also use Hamiltonian replica exchange to enhance the sampling, by selecting Use lambda hopping to enhance sampling.

Analyzing Simulations

5.1 Viewing Trajectories

You can play through trajectories, examine individual frames, and export trajectory data in a variety of forms, in the Trajectory panel. To open the Trajectory panel, click the T button in the Title column of the Project Table for the entry whose trajectory you want to view.

If you have entries in the Workspace, a panel opens asking if you want to keep them in the Workspace while the trajectory is played, or to remove them. Keeping the entries in the Workspace allows you to view the trajectory against a fixed background. You can superimpose the trajectory atoms on the background, and you can set up measurements, such as H-bonds, between the background atoms and the trajectory atoms, which are updated during play.

The toolbar in the Trajectory panel contains a standard set of controls for playing through the trajectory frames, which are listed below. The menu bar has one menu, Play, which contains items that correspond to the toolbar buttons.



Go to start
Display the first frame.



Previous
Display the previous frame.



Play backward
Display the frames in sequence, moving toward the first.



Stop
Stop playing through the frames.



Play forward
Display the frames in sequence, moving toward the last.



Next
Display the next frame.



Go to end
Display the last frame.



Loop

Choose an option for repeating the display of the frames. **Single direction** displays frames in a single direction, then repeats. **Oscillate** reverses direction each time the beginning or end of the frame set is reached.

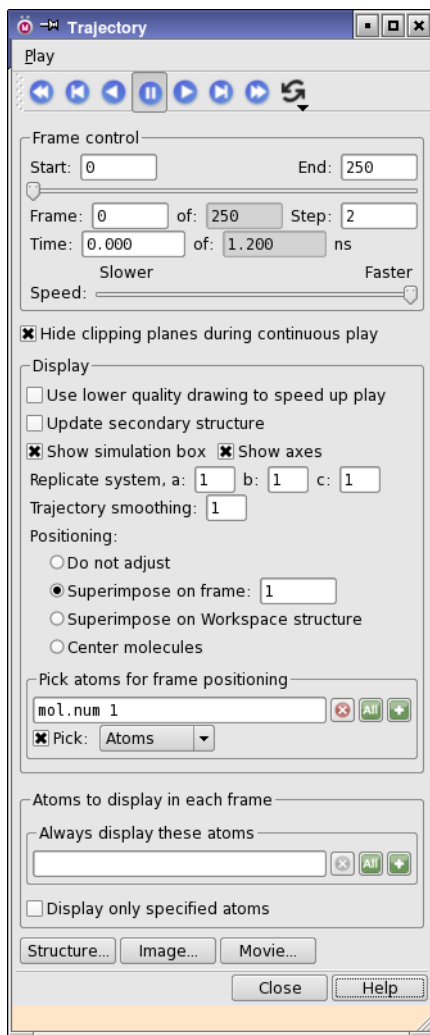


Figure 5.1. The Trajectory panel.

You can control the selection of frames and the speed of play in the Frame control section of the panel.

- The **Start** and **End** text boxes define the frames at which play starts and ends. Frames are numbered from 0.
- The **Frame** slider and frame text box can be used to select the frame to view. The current frame number is displayed in the text box below the slider. The total number of frames is also displayed in a noneditable text box.
- The **Step** text box sets the number of frames to step when playing through frames. This value does not affect the **Frame** slider. The frames that are selected for play can be exported as a selection of frames, using the output buttons.
- The **Time** text boxes display the time for the current frame and the total time for the trajectory. You can enter a time in the text box to select a frame.
- The **Speed** slider sets the speed at which the frames are played.

By default, the clipping planes window is automatically hidden when you play through a trajectory, as this speeds up play by about 50%. It is displayed again (if it was originally displayed) when play stops. If you want to see the clipping planes window during play, deselect **Hide clipping planes during continuous play**.

In the **Display** section you can control how frames are displayed in the **Workspace** and what features are displayed:

- **Use lower quality drawing to speed up play**—Use a lower quality representation of objects (tubes, spheres) in the **Workspace** to speed up play. This option has no effect on wire frame representation.
- **Update secondary structure**—Update the secondary structure assignment for each frame.
- **Show simulation box**—Show the edges of the simulation box (in purple).
- **Show axes**—Show the coordinate axes in green.
- **Replicate system**—Enter the number of replicas of the system to display in each of the three directions. This enables you to visualize the movement across the simulation box boundaries. These text boxes are unavailable if there are no periodic boundary conditions.
- **Trajectory smoothing**—Smooth the trajectory by averaging the coordinates over the specified number of frames.
- **Positioning**—Select on of these options to control the positioning of each frame relative to the **Workspace** during play.
 - **Do not adjust**—Do not adjust the positioning of each frame.

- **Superimpose on frame**—Align the structure in each frame by superimposing a selection of atoms on the corresponding atoms in the frame number given in the text box. Use the Pick atoms for positioning picking controls to make the atom selection.
- **Superimpose on Workspace structure**—Align the structure in each frame by superimposing a selection of atoms on the corresponding atoms in the Workspace (that are not part of the trajectory). Use the Pick atoms for positioning picking controls to make the atom selection, which must match both the Workspace structures and the trajectory structure.
- **Center molecules**—Center the selected molecules in the Workspace. Use the Pick atoms for positioning picking controls to make the atom selection.
- **Pick atoms for positioning**—Use these picking controls to select the atoms to superimpose or to center. You should consider picking atoms that do not change their position much during the simulation.

If you want to superimpose the trajectory on the Workspace structure, you must take care when you select the atoms to superimpose. The ASL expression for the selection in the Workspace structure is applied to each trajectory frame. If the ASL expression depends on the numbering (atom number, molecule number, etc.) and the order of the objects in the trajectory frames is not the same as in the Workspace structure, you may get unexpected results. You should use ASL expressions that are not order-dependent. You can use the atom selection button to choose from a variety of structural features whose ASL expressions are not order-dependent, like Backbone or Ligands. If the ordering is a problem, you could instead align the Workspace structure to a particular frame, and then play the trajectory by superimposing on that frame.

The atoms that are visible in each frame can be set either with the Workspace toolbar buttons, or with the tools in the Atoms to display in each frame section.

To select the atoms that are visible when frames are displayed, use the toolbar buttons that control the atom display:



The choice that you make with the toolbar buttons is recorded as an atom set, and is applied to each frame. The same atoms are always displayed in each frame, no matter where they move in the trajectory.

To display the atoms that come within a given distance of a particular set of atoms (such as a ligand or a binding site), use the Atoms to display in each frame section. You can select the atoms that are always visible with the Always display these atoms selection tools. You can then choose to display entire residues that have any atoms within a specified distance of these atoms. This expression is evaluated for each frame, which allows residues (such as water) to

move in and out of this distance. The particular set of atoms that is visible can therefore change from frame to frame.

The output buttons allow you to export the trajectory data in various forms. You can export individual frames, all frames, or the selection of frames defined by using the Start, End, and Step text boxes. The buttons have the following actions:

- **Structure**—Save structures from the trajectory to a file or create project entries from the structures. Opens the Export Structure dialog box, in which you can specify where the structures will go, which structures to export, and which atoms to export.
- **Image**—Create an image of the Workspace with the current frame displayed. Opens the Save Image panel.
- **Movie**—Save a movie of the trajectory in MPEG format. Opens the Export Movie panel, in which you can select the frames to be exported, the speed and the resolution.

If you want to view a trajectory while a simulation is running, you can do so by importing the *jobname-out.cms* file for the simulation from the host on which the simulation is running. This file contains information on the location of the trajectory, and should be in the temporary directory for the job. There may also be a copy of the output CMS file in the working directory on the local host, but the trajectory will not be present until the job finishes. Once you have imported the file, click the T button for the imported entry. To update the trajectory later, you must reimport the file.

5.2 Simulation Quality Analysis

The Simulation Quality Analysis panel displays a simulation summary and an analysis of the total energy, potential energy, temperature, pressure, and volume over the length of the simulation. The analysis includes the average value, the standard deviation, and the slope of a linear fit to the values of the property as a function of time. You can run the analysis on a completed simulation or while the simulation is running.

To open the Simulation Quality Analysis panel, choose Applications → Desmond → Simulation in the main window.

To load the desired simulation, click Browse, and navigate to the desired Energy Sequence file for the simulation, which has a .ene suffix. If you want to perform the analysis on a running simulation, you must use the file from the host on which the simulation is running.

The analysis performs averaging of the results over short time periods (“blocks”) to reduce the noise and eliminate correlation between consecutive reports. To change the size of the block, enter a value in the Block length for averaging text box.

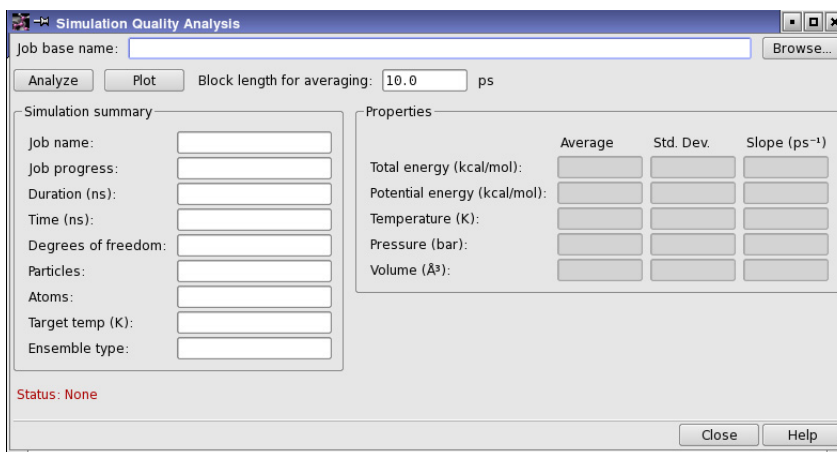


Figure 5.3. The Simulation Quality Analysis panel.

When you have the desired block length, click **Analyze** to perform the analysis. The analysis can take a few minutes. When it finishes, the **Simulation summary** and **Properties** sections are filled in with the results of the analysis. If you want to view a plot of the thermodynamic properties as a function of simulation time, click **Plot**. A panel is displayed with the results plotted.

5.3 Simulation Event Analysis

You can use the Simulation Event Analysis panel to analyze the output from a Desmond simulation and display the results of the analysis in various forms. The analysis involves the selection of properties that are extracted from the trajectory; these properties can then be exported and plotted, and statistics calculated for the properties.

To open this panel, choose **Applications → Desmond → Simulation Event Analysis**. Detailed information about using this panel is available in the online help, which you can open by clicking the **Help** button in the panel. A summary is given below.

Five types of properties are available: energy, hydrogen bonds, RMS deviation of atoms relative to a trajectory frame or reference structure, RMS fluctuation of atoms over the entire trajectory relative to a trajectory frame or reference structure, and measurements of distances, angles, dihedrals, and radius of gyration.

The first task in the analysis is to select properties and the atoms or molecules for which they are calculated. These are added to the list in the center of the panel, where they are represented as “keywords”. Next, these properties must be evaluated, by selecting individual properties in the list and clicking **Analyze**, or clicking **Analyze All** to analyze all the properties. You can read or write list of properties (keywords), and you can export the results of the analysis.

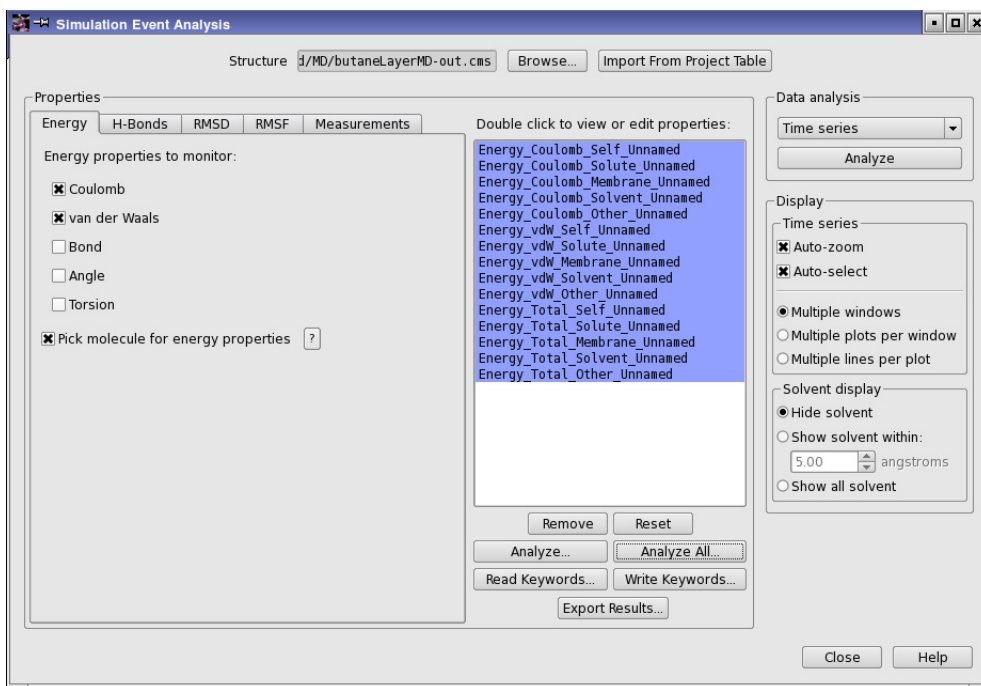


Figure 5.4. The Simulation Event Analysis panel.

Once the properties have been evaluated, you can display the results in various kinds of plots: time series, histogram, multi-variable plot, heat map, polar plot, RMSF plot, and also display statistics on the properties.

5.4 Replica Exchange Review

The Replica Exchange Dynamics Review panel can be used to display a plot of the replicas as a function of time, on a temperature versus time graph. Each replica is colored uniquely so you can trace its temperature as a function of time.

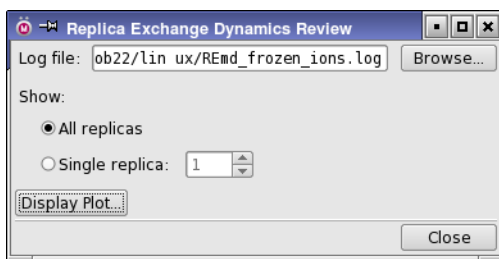


Figure 5.5. The Replica Exchange Dynamics Review panel.

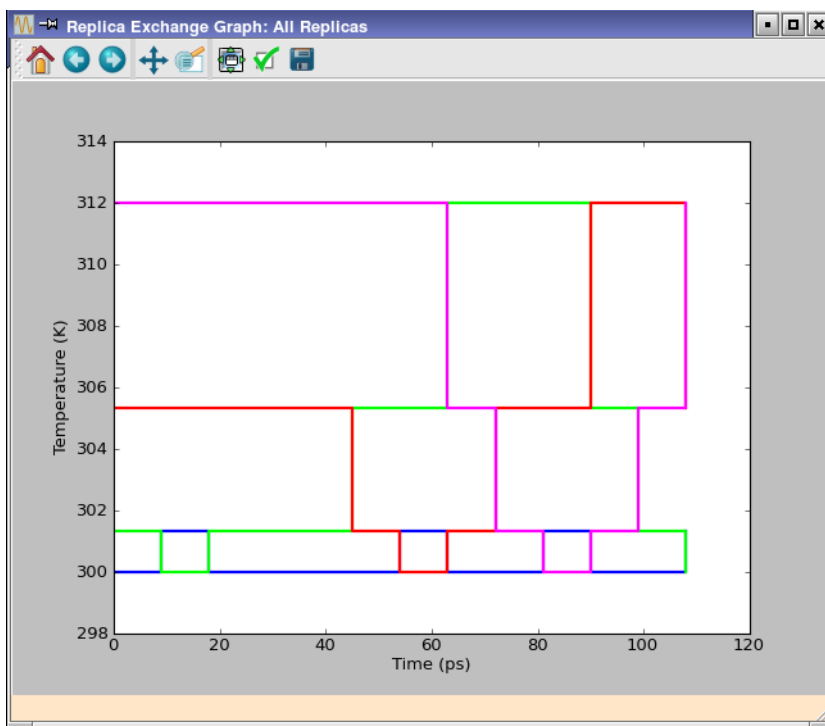


Figure 5.6. The Replica Exchange Graph panel.

To open this panel, choose Applications → Desmond → Replica Exchange Review or Tasks → Molecular Dynamics → Replica Exchange Review. First, you must load a log file for the simulation (*not* the multisim log file). Then you can choose whether to show all replicas or a single chosen replica. When you have made a choice, click Display Plot. The plot is displayed in a new window, with a standard set of plotting controls for configuring the plot and saving an image. If you are displaying single replicas, you can add replicas to the plot by selecting a new one, or you can plot a different replica by closing the plot window, selecting a new replica, and clicking Display Plot again.

5.5 Metadynamics Analysis

The Metadynamics Analysis panel analyzes a metadynamics calculation and displays a plot of the free energy as a function of the collective variables (CVs) used for the calculation. You can analyze calculations with one or two collective variables. The free energy value can be plotted relative to the minimum value or as an absolute value. The plots can be saved and loaded. To open this panel, choose Applications → Desmond → Metadynamics Analysis.

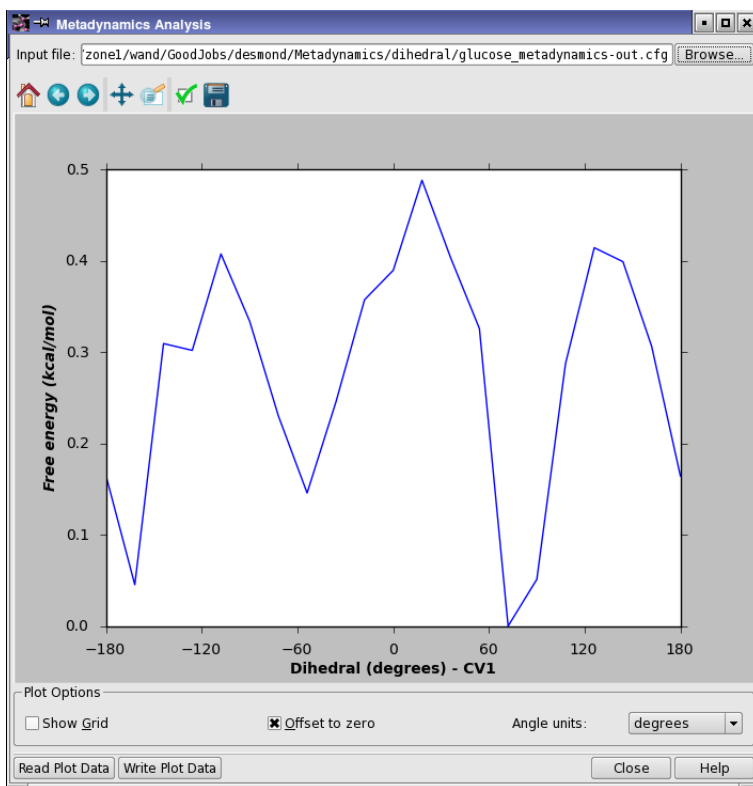


Figure 5.7. The Metadynamics Analysis panel.

Running Simulations from the Command Line

Desmond jobs may be started from Maestro or from the command-line. The mechanisms for running jobs from the command line are described in this chapter. You might wish to run Desmond from the command line for any of the following reasons:

- To exercise greater control over Desmond behavior
- To debug an aberrant run
- For convenience
- To transfer the job to a remote location

The Desmond panels can be used to write out (using the Write button) `.cfg` files for a simulation. Those `.cfg` files can then be run elsewhere, routinely from a batch script or customized prior to use. More information on the contents of `.cfg` files is available in [Appendix B](#).

Desmond jobs run under Schrödinger's Job Control facility. This facility manages the execution and monitoring of jobs, and handles the input and output files and the incorporation of results into a Maestro project. The [Job Control Guide](#) describes how to set up the information needed for Job Control to run on the computers to which you have access. It includes information on remote hosts, clusters, and batch queues.

As is the case for all Schrödinger software, the environment variable `SCHRODINGER` must be set to the directory where the Schrödinger software, including Desmond, was installed. In addition, there are other environment variables that can be set to override default resource values. See [Appendix B](#) of the *Job Control Guide* for more information.

By default, the Schrödinger job control facility uses `ssh` to communicate between remote nodes. For more information, see [Section 7.2](#) of the *Installation Guide*.

This chapter describes the command syntax for running single-stage and multistage Desmond jobs, and for building the input composite model system. For details of the file formats and technical background, see the [Desmond User's Guide](#), from D. E. Shaw Research.

6.1 The desmond Command

Desmond is run from the command-line by executing the script `desmond`, which is located in the `$SCHRODINGER` directory. The `desmond` script determines the proper executables to run for the host being used. Additional utility programs are provided in the `$SCHRODINGER/utilities` directory.

The syntax for the `desmond` command is as follows:

```
$SCHRODINGER/desmond [options] -in cms-file -c config-file
$SCHRODINGER/desmond [options] -restore checkpoint-file
```

Use the `-in` option for a new job, and the `-restore` option for restarting a job. The input file for a new job must be a CMS file, with extension `.cms`, and a configuration file (`.cfg`) must also be specified. For restarting a job, a checkpoint file, with extension `.cpt`, must be provided.

The options for the `desmond` command are listed in [Table 6.1](#). The standard Job Control options are also supported—see [Section 2.3](#) of the *Job Control Guide*. Additional diagnostic options, which do not run the job but provide information, are also given in the section mentioned. In particular, you should note the syntax of the `-HOST` option, which is used to specify the list of hosts used for the job.

For information on storage of temporary files, interacting with running Desmond jobs, and so on, see the *Job Control Guide*.

Table 6.1. Options for the `desmond` command.

Option	Description
<i>Job Options</i>	
<code>-DEBUG</code>	Print diagnostic output.
<code>-LOGINTERVAL</code>	Interval for copying the log file back to the submission host. Default: 5s.
<code>-INTERVAL</code>	Interval for copying monitor files back to the submission host. Default: 5s.
<code>-p -P -PROCS -NPROC</code>	Number of processors to use. Default: 1.
<code>-JOBNAME name</code>	Specify the name for the job. The default is the input <code>.cfg</code> or <code>.cpt</code> file name, minus the extension.
<code>-jin filename</code>	Files to be transferred to the execution host by Job Control.
<code>-jout filename</code>	Files to be copied back to the submission host by Job Control.
<code>-h[elp] -HELP</code>	Print usage information
<code>-v</code>	print version information and exit

Table 6.1. Options for the *desmond* command. (Continued)

Option	Description
<i>Program Options</i>	
<code>-exec program</code>	Run the specified Desmond program. Available programs are: <code>mdsim</code> —Run a molecular dynamics simulation (the default) <code>minimize</code> —Run a minimization <code>remd</code> —Run a replica exchange molecular dynamics simulation <code>vrund</code> —Analyze a trajectory If a front-end config file is provided with <code>-c</code> (such as those created by Maestro), the program to use is determined automatically and this option is not needed.
<code>-c config-file</code>	Parameter file for simulation. Required with <code>-in</code> for a new simulation.
<code>-cfg key=val</code>	Specify extra simulation parameters. Multiple instances of this option can be supplied to specify multiple extra parameters.
<code>-comm plugin</code>	Use the specified communication plugin. Available plugins are <code>serial</code> and <code>mpi</code> . Default: <code>serial</code> .
<code>-dp</code>	Run double-precision version. Default: run single-precision version.
<code>-noopt</code>	Do not optimize parameters automatically.
<code>-t temperature</code>	Specify temperature in a replica exchange MD simulation. Include an instance of this option for each temperature in the exchange.
<code>-tpp n</code>	Specify the number of threads per processor.

Some examples of running *desmond* from command line are shown below:

1. Running a 4-processor job on a queuing system or a particular machine (myhost):

```
$SCHRODINGER/desmond -HOST myhost -P 4 -c x.cfg -in x.cms
```

2. Running a 4-processor job on two different machines (host1 and host2):

```
$SCHRODINGER/desmond -HOST "host1:2 host2:2" -P 4 -c x.cfg  
-in x.cms
```

3. Continuing a 4-processor simulation from a checkpoint file (x.cpt):

```
$SCHRODINGER/desmond -HOST myhost -P 4 -restore x.cpt
```

4. Running a replica exchange simulation, in which each replica uses 2 processors with different input CMS files:

```
desmond -P 4 -c x.cfg -t 300 -in x300.cms -t 310 -in x310.cms
```

6.2 Running Multiple Simulations

Molecular dynamics studies usually involve performing a series of calculations. For example, a sequence of minimization, thermalization, and relaxation calculations followed by the production simulation itself is usually undertaken for molecular dynamics study. The MultiSim utility (`multisim`) facilitates the task of running multiple Desmond-related or MCPRO⁺-related calculations in a sequential manner. The Desmond general and FEP panels can be used to write out `.msj` files for those tasks. These `.msj` files can then be run elsewhere, for example from a batch script, or customized prior to use. More information on the contents of `.msj` files is available in the Maestro online help—see the list of Desmond topics.

The syntax for the `multisim` utility is as follows:

```
$SCHRODINGER/utilities/multisim [options] [-i] structure-file -m msj-file
$SCHRODINGER/utilities/multisim [options] -RESTART checkpoint-file
```

The first syntax should be used for new jobs, and the second for restarting jobs. The `-i` for the structure file is deprecated, but can still be used. The structure file can be either a Maestro (`.mae`) file or a CMS (`.cms`) file. For restarting jobs, `-r` can be used instead of `-RESTART`, but `-r` is also deprecated. The options are described in [Table 6.2](#), including job options. The standard Job Control options are also supported, as are the options `-LOCAL`, `-WAIT`, and `-NOJOBID`—see [Section 2.3](#) of the *Job Control Guide*.

Table 6.2. Options for the `multisim` utility.

Option	Description
<code>-ADD_FILE filename</code>	Additional input file to copy to the working directory of the <code>multisim</code> job. By default, <code>multisim</code> identifies and copies most files needed for the run.
<code>-c cfg-file</code>	File that contains the default Desmond config parameters. This option is only useful for Desmond subjobs.
<code>-cpu num-cpus</code>	Number of processors for each Desmond subjob. This can be a string indicating the processor topology e.g., '2 2 2'. If this option is not provided, a default value set by either the backend or the protocol will be used. Processor counts must be a power of 2, 3, or 5, or products of these powers.
<code>-d stage-file</code>	File (<code>.tgz</code>) containing information on the stages from the previous job, when restarting a simulation. Use one instance for each file. Requires a <code>multisim</code> checkpoint file.
<code>-debug</code>	Turn on <code>multisim</code> debug mode.
<code>-DEBUG</code>	Turn on both <code>multisim</code> and Job Control debug modes.

Table 6.2. Options for the *multisim* utility. (Continued)

Option	Description
<code>-description string</code>	Job description to print to the log file.
<code>-h[elp], -HELP [topic]</code>	Print usage message and exit. You can specify an optional topic with the following topic references: @all—Show concise help of all valid options. @doc—Show detailed help of all valid options. @option—Show detailed help of the specified option, e.g., @-cpu. @examples—Show a few command examples. @deprecated—Show a list of deprecated options.
<code>-SUBHOST subjob-host</code>	Host on which to run the subjobs. <code>-host</code> can also be used instead of <code>-SUBHOST</code> , but is deprecated. This option does not support the <i>host:cpus</i> syntax.
<code>-HOST multisim-host</code>	Host to run the <i>multisim</i> master job.
<code>-JOBNAME jobname</code>	Job name of this <i>multisim</i> job
<code>-RETRIES maxretries</code>	Maximum number of times to restart failed subjobs. <code>-max_retries</code> can also be used instead of <code>-RETRIES</code> , but is deprecated.
<code>-maxjob maxjob</code>	Maximum number of simultaneous subjobs
<code>-mode mode</code>	Run in non-default job management mode. The only allowed value is <i>umbrella</i> , for which subjobs run on the same node as the master job, <code>-SUBHOST</code> is ignored and <code>-maxjob</code> is set to 1. All processors are assigned to the master job, so that the subjobs do not need to be queued. You must use an MPI-parallel queue with this option.
<code>-nopref</code>	
<code>-notify email</code>	Send email notification to the specified address when the job finishes.
<code>-o output-file</code>	Output <i>.cms</i> or <i>.mae</i> file. This file can be incorporated into <i>Maestro</i> .
<code>-probe checkpoint-file</code>	Probe a checkpoint file and quit. The state of the job recorded in the checkpoint file is displayed, and whether the checkpoint file is compatible with the current version of <i>multisim</i> .
<code>-quiet</code>	Light log information (default).
<code>-set string</code>	Specify parameters that take precedence over those in the <i>.msj</i> file, as a quoted string. Can be specified multiple times to make more than one change to the parameters.
<code>-v[ersion]</code>	Print the <i>multisim</i> version and exit.
<code>-verbose</code>	Heavy log information

6.2.1 Examples of Running MultiSim

A simple `multisim` command can take the form:

```
$SCHRODINGER/utilities/multisim -i input.cms -m input.msj
    -HOST mycluster -JOBNAME multistage
```

which runs the `multisim` utility with the starting structure from `input.cms`. The instructions for different stages of `multisim` are provided in `input.msj`. The master job, which coordinates the subjobs, is run on the host labeled `mycluster` in the `schrodinger.hosts` file. The job name is `multistage`.

The following command explicitly uses a different host for the master job and the subjobs:

```
$SCHRODINGER/utilities/multisim -i input.mae -m input_fep.msj
    -HOST tabitha -host kyla_para -JOBNAME example_fep_job
    -maxjob 6 -cpu "2 2 2"
```

The `-HOST` option specifies the host of the master job, whereas `-host` specifies the compute host for the subjobs. The keyword `-maxjob 6` means that at most 6 subjobs are simultaneously in the queue of the compute host. When one of the subjobs finishes the next subjob is launched on the compute host. At any time, the number of the subjobs launched from the master job does not exceed 6 on the compute host.

6.2.2 Sample MultiSim Job (.msj) File

An example `.msj` file is given below. The file syntax is described in full in [Appendix A](#). The file is modular, consisting of a series of stages, each of which contains a list of keywords and value settings enclosed within braces. Lines beginning with `#` are comments and hence are ignored.

```
# Desmond standard NPT relaxation protocol
# All times are in the unit of ps.
# Energy is in the unit of kcal/mol.
task {
    task = "desmond:auto"
}

minimize {
    title           = "Minimization with restraints on solute"
    max_steps       = 2000
    steepest_descent_steps = 10
    convergence     = 50.0
    restrain        = { atom = solute force_constant = 50.0 }
}

minimize {
```

```

title           = "Minimization without any restraints"
max_steps       = 2000
steepest_descent_steps = 10
convergence     = 5.0
}

simulate {
  title      = "Berendsen NVT, T = 10 K, small timesteps, and restraints on solute
heavy atoms"
  annealing  = off
  time       = 12
  timestep   = [0.001 0.001 0.003]
  temperature = 10.0
  restrain   = { atom = solute_heavy_atom force_constant = 50.0 }
  ensemble   = {
    class = NVT
    method = Berendsen
    thermostat.tau = 0.1
  }

  randomize_velocity.interval = 1.0
  eneseq.interval            = 0.3
}

simulate {
  title      = "Berendsen NPT, T = 10 K, and restraints on solute heavy atoms"
  annealing  = off
  time       = 12
  temperature = 10.0
  restrain   = retain
  ensemble   = {
    class = NPT
    method = Berendsen
    thermostat.tau = 0.1
    barostat .tau = 50.0
  }

  randomize_velocity.interval = 1.0
  eneseq.interval            = 0.3
}

simulate {
  title      = "Berendsen NPT and restraints on solute heavy atoms"
  effect_if  = ["@*.*.annealing"] 'annealing = off temperature =
"@*.*.temperature[0][0]''
  time       = 12
  restrain   = retain
  ensemble   = {
    class = NPT

```

```
        method = Berendsen
        thermostat.tau = 0.1
        barostat .tau = 50.0
    }

    randomize_velocity.interval = 1.0
    eneseq.interval             = 0.3
}

simulate {
    title           = "Berendsen NPT and no restraints"
    effect_if       = [["@*.*.annealing"] 'annealing = off temperature =
"@*.*.temperature[0][0]''']
    time            = 24
    ensemble        = {
        class       = NPT
        method      = Berendsen
        thermostat.tau = 0.1
        barostat .tau = 2.0
    }

    eneseq.interval = 0.3
}

simulate {
    cfg_file = "example.cfg"
    jobname  = "$JOBNAME"
    dir      = "."
    compress = ""
}
```

In the above example, the first stage, `task`, specifies the type of job that is run so that appropriate defaults can be set. In this case `desmond:auto` indicates that it is a Desmond job and that the type of job should be detected automatically.

The second stage is a minimization of the system over a maximum of 2000 steps. Of the 2000 steps, the first 10 steps are steepest descent. The convergence criterion is set rather loosely to 50.0 kcal mol⁻¹Å⁻¹. The solute is restrained with a force constant of 50.0 kcal mol⁻¹Å⁻¹. The monitor file is not updated. The third stage is similar except that nothing is restrained.

The fourth stage sets up a 12 ps Berendsen NVT simulation. For this simulation, the temperature is set to 10.0 K, and the thermostat relaxation is set to 0.1 ps. Resampling is done every 1 ps. The solute atoms are restrained. Checkpointing and structure monitoring are turned off. Center-of-mass motion is removed and the `.ene` file is updated more frequently, at intervals of 0.3 ps. Subsequent simulation stages follow with progressively more freedom until the second last stage when conditions resemble the production run.

The last stage is the production dynamics simulation. The simulation parameters could have been explicitly listed here as in the preceding stages. In this example, this stage simply refers to a Desmond .cfg file. The last line gives an example of how this .msj file could be run. You would need to change the options for the job.

More examples can be found in the `$SCHRODINGER/mmshare-vversion/data/desmond` directory. Detailed information on the syntax of .msj files is available in [Appendix].

6.2.3 Treatment of Intermediate Files

In general, the files from each stage are copied back to the launch directory as a gzipped tar archive with the name `jobname_stage-number.tgz`. The production simulation stage is treated differently and the files are directly copied back without being archived. This treatment would be similar to a simple molecular dynamics simulation (i.e. not a multisim job). This allows the final output CMS file to be incorporated and the trajectory to be played as a normal job.

6.3 Building a Model System

To prepare a model system from the command line, you can use the system builder utility, `system_builder`. You can choose to add solvent, ions, and insert the solute into a membrane. The `system_builder` utility takes a composite system builder (CSB) file as input. This file contains all the required information to convert a solute file into a solvated system.

The syntax of the `system_builder` command is as follows:

```
$SCHRODINGER/utilities/system_builder [-rezero] [-minimize_volume]
    [job-options] input.csb
```

The standard Job Control options, described in [Table 2.1](#) of the *Job Control Guide*, are accepted, as are the options `-WAIT`, `-LOCAL`, and `-NOJOBID`, described in [Table 2.2](#) of the *Job Control Guide*.

The possible steps involved in the conversion are:

- Reading the solute from a file
- Reading the water model
- Reading the positive ion data
- Reading the negative ion data
- Adding a membrane
- Solvating
- Neutralizing
- Writing out a composite molecular system (CMS) file.

Two other global tasks are performed by setting the command-line options:

- rezero Set the coordinate origin to the centroid of the solute coordinates.
- minimize_volume Minimize the volume of the simulation box.

The order of the keywords in a CSB file matters. Lines beginning with # are considered to be comments and are ignored. An example CSB file is shown below:

```
{
  read_solute_structure mysolute_setup-in.mae    # solute file name
  solvent_desmond_oplsaa_typer {
    input_file_name spc.box.mae
    run
  }
  positive_ion_desmond_oplsaa_typer {
    input_file_name Na.mae
    run
  }
  negative_ion_desmond_oplsaa_typer {
    input_file_name Cl.mae
    run
  }
  membranize POPE.mae.gz 10.000000 10.000000
  create_boundary_conditions orthorhombic 0.000000 0.000000 10.000000
  exclude_ion_from { 1 2 } 10.0
  solvate
  neutralize
  write_maeff_file chorus_setup-out.cms
}
```

The various sections of the file are described in the following subsections.

6.3.1 Reading the Structures

The first five sections read in all the structural information for the solute, solvent, and ions. To locate the files that contain this information, the current directory is searched first, then the directory \$SCHRODINGER/mmshare-vversion/data/system_builder.

The first section reads the solute and solvent structure.

```
read_solute_structure mysolute_setup-in.mae    # solute file name
solvent_desmond_oplsaa_typer {
  input_file_name spc.box.mae
  run
}
```

The keyword `read_solute_structure` reads the solute structure from the specified file. If the file is given as a relative path, it is copied to the temporary directory for the job. If it is given as an absolute path, the file must exist at that location on the execution host. The structures can be a solute structure, structures for different stages of FEP simulations, or a completely solvated system. The force-field information is obtained by either reading from an existing `ffio_ff` block in the input file or running the force field server.

Note that alternate coordinates are removed from the solvent structure when it is imported.

The `solvent_desmond_oplsaa_typer` section describes the solvent system that is used to solvate the structure. The keyword `input_file_name` specifies the file to be used to read the solvent model.

The next two sections determine the data to be used for counter ions to neutralize (or in some cases charge) the system.

```
positive_ion_desmond_oplsaa_typer {
    input_file_name Na.mae
    run
}
negative_ion_desmond_oplsaa_typer {
    input_file_name Cl.mae
    run
}
```

These two sections describe the ion systems to neutralize or to add ions to the current structures. In each section one `input_file_name` keyword is provided to determine the file to be used.

The next two sections (not used in the example file above) similarly determine the ion systems to add salts to the system.

```
salt_positive_ion_desmond_oplsaa_typer {
    input_file_name Na.mae
    run
}
salt_negative_ion_desmond_oplsaa_typer {
    input_file_name Cl.mae
    run
}
```

The syntax for this section is same as for the positive and negative ion sections.

6.3.2 Adding a Membrane

The keyword `membranize` instructs the system builder to create a membrane patch around the current solute structure.

```
membranize filename x-buf y-buf
```

The first input parameter is the name of the file that includes the membrane template and its equilibrating solvent. The other two parameters *x-buf* and *y-buf* specify the minimum distance between the solute and the box boundary in the plane of the membrane.

6.3.3 Setting the Box Shape and Dimensions

The box shape and dimensions can be specified either in terms of an absolute size or in terms of a buffer distance.

The keyword `boundary_conditions` is used to define a box with specified absolute dimensions. The command using this keyword can be one of the following:

```
boundary_conditions cubic a
boundary_conditions orthorhombic a b c
boundary_conditions triclinic a b c alpha beta gamma
```

The box shape is the first parameter, and can be `cubic`, `orthorhombic`, or `triclinic`. It is followed by the absolute size of the box defined by *a* or by *a*, *b*, and *c*. For a triclinic box shape, the angles *alpha*, *beta*, and *gamma* must also be given.

Alternatively, the keyword `create_boundary_conditions` can be used to specify a box in terms of the distance between solute atoms and box edges. The command using this keyword can be one of the following:

```
create_boundary_conditions cubic a
create_boundary_conditions orthorhombic a b c
create_boundary_conditions triclinic a b c alpha beta gamma
```

The parameters are similar to those for the absolute box size, but the distances are the minimum distance any solute atom and the box boundary in the given direction. The distance between two images of the solute structures is therefore twice the specified values. If a membrane is used, the box shape must be `orthorhombic` and the *a* and *b* values must be set to zero so that solvent molecules are not placed within the membrane layers.

You can minimize the size of the box by using the `-minimize_volume` option to the `system_builder` command.

6.3.4 Setting Force Field Information

The command `set_oplsaa_version` can be used to specify OPLSAA version. Currently only OPLS_2005 (value: 2005) is supported.

```
set_oplsaa_version 2005
```

By default, force-field parameters that are already present in the solute CTs (the `ffio_ff` block) are replaced. The following setting can be used to retain existing parameters:

```
remove_solute_ffio no
```

6.3.5 Setting the Number and Location of Ions

The addition of ions to the system is governed by several keywords: `add_ion`, `ion_location`, `exclude_ion_from`, `add_salt`, and `neutralize`.

Two commands using the keyword `add_ion` can be used to add specific numbers of positive and negative ions.

```
add_ion positive 5
add_ion negative 5
```

The keyword `add_ion` adds positive or negative ions to the system. The number of ions to be added is specified as the second argument of this keyword.

The `ion_location` keyword can be used to specify the proximity of ions with reference to certain atoms.

```
ion_location { { atom1 } { atom2 } }
```

The keyword `ion_location` is followed by the list of solute atom indexes. Ions are placed near the listed atoms. For each ion, the atom index is given inside braces. Extra atom index specifications are ignored. If there are fewer atom index specifications than there are ions, the remaining ion locations are determined randomly.

In order to exclude ions near certain atoms, the `exclude_ion_from` keyword can be used:

```
exclude_ion_from { atom-list } distance
```

The keyword `exclude_ion_from` has two arguments. The first argument is a list of atom indices of solute atoms, in braces. The second argument is the distance value. No ion, whether a single ion or from a salt, is placed within the given distance in angstroms from the listed atoms.

The salt concentration is determined using the `add_salt` keyword.

```
add_salt conc
```

The keyword `add_salt` adds the positive and negative salt ions to the system defined above. The argument specifies the salt concentration for the molecular system. Based on the volume of the solvent and the concentration, the number of salt ions is calculated and rounded to an integer value. The coordinates of salt ions are determined randomly.

The following command can be used to neutralize the system.

```
neutralize
```

The net charge of the current solute structures is calculated and the number of counter ions necessary to neutralize the system is added.

6.3.6 Solvating the System

This command is used to initiate the solvation of the current solute structure, which includes the ions and the membrane.

```
solvate
```

The keyword `solvate` is used to solvate the current solute structures using the solvent system defined in the `solvent_desmond_oplsaa_typer` section. The solvent system is extended to be consistent with the boundary condition defined by the `boundary_conditions` or `create_boundary_conditions` section. Any solvent molecules that overlap the solute structures are removed.

6.3.7 Writing the Output File

This command is used to write the output to a “maeff” (Maestro + force-field) file, otherwise known as a *composite model system* (CMS) file, which has the extension `.cms`.

```
write_maeff_file filename
```

The keyword `write_maeff_file` writes the current composite structures and their force field parameters into a file in CMS format. The first structure contains all of the molecules in the system and is usually referred to as the “full system CT”. The structures (or CTs) that follow it contain different components of the system. For instance there usually is a solute structure, a solvent structure (containing all of the solvent molecules) and structures for different types of ions. There may also be a structure that contains solvent molecules that are extracted from the solute. The force field parameters are inserted as `ffio_ff` blocks within each CT block (“structure”) except the first. The coordinate origin of the structures is by default the center of mass of the solutes.

Using VMD for Desmond Trajectories

VMD [7] is a powerful program for visualizing molecular dynamics simulations that is available from the Theoretical and Computational Biophysics Group at the University of Illinois at Urbana-Champaign. A plugin for Desmond is provided as part of VMD that makes it possible for VMD to read and write Maestro files and read Desmond trajectories. The output Maestro files are suitable for use in building model systems that can then be used to run Desmond.

This chapter contains information on reading a CMS file and a Desmond trajectory into VMD, and writing a Maestro file from VMD. For information on installing VMD, see [Section 3.10.4](#) of the *Installation Guide*.

To start VMD, enter the command:

```
vmd
```

Two windows are opened, VMD main and VMD *version* OpenGL Display. The former can be used to control VMD while the latter can be used to display the molecular systems and to view trajectories. VMD has extensive documentation which is available from the Help menu in the VMD main window.

7.1 Reading a CMS File and a Desmond Trajectory

This section describes how to read a Maestro CMS file and a Desmond trajectory into VMD. To view a Desmond trajectory in VMD, you must first read in the output structure CMS file from a Desmond simulation. This file is usually named *jobname*-out.cms. Viewing trajectories from Desmond FEP simulations is not currently supported.

To read in a CMS file:

1. Choose New Molecule from the File menu of the VMD Main window.

The Molecule File Browser panel opens.

2. Click Browse.

A file selector opens.

3. Navigate to and select the output .cms file from a Desmond simulation, and click OK.

The file selector closes.

4. In the Molecule File Browser panel, set the file type to Maestro File (no timesteps).
5. Click Load.

You have just created a Molecule listing in VMD which should appear as a new a line in the VMD Main window corresponding to the CMS file that you just read in.

To read in a trajectory for this system:

1. Ensure that the line for the CMS file that you just read in is the only one that is highlighted.
2. Choose Load Data into Molecule from the File menu in the VMD Main window.

The Molecule File Browser panel opens.

3. Click Browse.

A file selector opens.

4. Navigate into the trajectory directory from a Desmond simulation.
5. Select the file `clickme.dtr` and click OK.

The file selector closes. In the Molecule File Browser the file type should now be listed as Desmond trajectory.

6. Click Load.

The trajectory should load into VMD and automatically start playing.

7.2 Writing a Maestro File

You can export individual configurations into a Maestro file from within VMD. This can be a useful way to convert configurations produced by various programs into a format that will function with Maestro.

To export a Maestro file from VMD:

1. Select the molecule that you want to export in the VMD main window.
2. Choose Save Coordinates from the File menu.

The Save Trajectory panel opens.

3. Set the File type to mae.
4. Click Save.

5. In the Filename text box, edit the directory, and append the name for the new file to the directory.

This is a plain Maestro file, whose extension should be .mae.

6. Click OK.

Alternate Force Field Parameters and Constraints

The Desmond installation includes two utilities, `viparr` and `build_constraints` that can be used to add or adjust the force field parameters and the accompanying constraints for chemical systems prior to simulating them with Desmond. Both programs read and write Maestro structure files.

Viparr is a template-based force field assignment utility that comes with a number of built-in force fields including some developed for Amber and CHARMM (see below for more information). User-defined force fields are also supported by viparr, if they are provided in viparr's file format. Viparr can be used to specify different force fields for various components of the system, provided that the force fields are compatible. This flexibility makes it possible to do some things that may be useful in force-field development including:

- Using one force field for one part of the chemical system and another force field for another part (this allows you, for example, to easily switch between water models)
- Using one or more components from one force field (e.g., the dihedral parameters) and the remaining components from another force field
- Overriding some of the parameters (e.g., some but not all of the angle parameters) with those from another force field

Some classes of constraints are often used with, and in some cases required by, various force field representations of molecules. The utility `build_constraints` adds these constraints to a structure file. You should run this utility after running `viparr`, to ensure that the force field is ready for use.

8.1 The viparr Utility

To run `viparr`, use the following command:

```
$SCHRODINGER/run -FROM desmond viparr.py [options] input-file output-file
```

where *input-file* and *output-file* are the input and output structure (CMS) files, respectively. This utility does not run under Job Control. It does not take much time, so it can be run locally.

The basic options are given in [Table 8.1](#). The force field data files are located in the directory `$SCHRODINGER/desmond-vversion/data/viparr`. When you have run `viparr`, you must run `build_constraints`, to ensure that the force field constraints are set properly.

Table 8.1. Options for the *viparr* utility.

Option	Description
-h	Print usage message, including the names of the built-in force fields
-n <i>name</i>	Specify force field name or other annotation to put into the output file. If not specified, a default name is used.
-f <i>ffname</i>	Specify a built-in force field. The available force fields are listed in Table 8.2 . You can repeat this option to specify multiple force fields, one per instance of the option. Parameters of force fields listed earlier override parameters of force fields listed later. When multiple force fields are specified, the order is important if some parameters are intended to override others.
-c <i>ctnum</i>	Specify the index of a single structure (“CT block”) in the input file for processing by <i>viparr</i> . Structures are numbered starting at 1. You can provide multiple -c options to specify more than one structure to process. The default is to process all structures.
-d <i>ffdir</i>	Specify a user-defined force-field directory. You can repeat this option to specify multiple directories. As for -f, the order is important.
-m <i>mergedir</i>	Path to user-defined force field directory that is to be merged with previously specified force field. Multiple directories can be specified with multiple instances of this option.
-nocompress	Do not use <i>ffio</i> block compression.
-p <i>pdir</i>	Specify the plugin directory. The default is to use the directory defined by the environment variable <code>VIPARR_PDIR</code> , which contains the standard plugins. All necessary plugins, including those for the built-in force fields, must be in the directory specified by -p.
-v	Verbose output.

The available force fields are listed in [Table 8.2](#), with references.

Table 8.2. Force fields available with viparr.

Force Field	Ref.	Force Field	Ref.
Amber		OPLS-AA	
amber94	8	oplsaa_impact_2001 ^a	24-29
amber96	9	oplsaa_impact_2005 ^{b,c}	24-31
amber99	10	oplsaa_ions_Jensen_2005	32
amber99SB	10,11	Water models	
amber99SB-ILDN	12	spc	34
amber03	13	spce	35
CHARMM		tip3p	36
charmm22nocmap	14-16	tip3p_charmm	37
charmm27	14-21	tip4p	38
charmm32	14,15,20-22	tip4pew	39
charmm36_lipids	23	tip4p2005	40
		tip5p	

- a. Parameter assignment as implemented in the OPLS_2001 force field in Impact.
b. Parameter assignment as implemented in the OPLS_2005 force field in Impact.
c. Note that the system_builder's implementation of OPLS_2005 has more general coverage of ligand-like molecules than viparr.

8.2 The build_constraints Utility

To create a constraint block use the following command:

```
$SCHRODINGER/run -FROM desmond build_constraints.py [options] input-file
output-file
```

where *input-file* is a structure file that has been previously processed with viparr and *output-file* is a new structure file for the system with the constraints added. The options are given in Table 8.3. The following constraint types are detected and automatically added:

- AH n , where n is a count of the number of hydrogen atoms (1, 2, 3, ...) in a group composed of a heavy atom and the hydrogen atoms directly bonded to it
- HOH, oxygen bonded to two hydrogen atoms and no other atoms.

Table 8.3. Options for the *build_constraints* utility.

Option	Description
-a, --angles	Constrain AH2 and AH3 angles.
-h, --help	Show usage message and exit.
-k, --keep	Keep bonded terms that coincide with constraints.
-r, --revert	Remove constraints and restore built terms.
-v, --verbose	Print constraint terms.
-x <i>exclude</i> , --exclude= <i>exclude</i>	Don't use the specified type of constraint (HOH, AH1, AH2, ...).

8.3 Input and Output Files

The input structure file should contain all the atoms in the chemical system that are to be simulated, including hydrogen atoms, water molecules, ions, and so on. The chemical system may contain a number of structures (also called connection tables or CTs). Residues (including water molecules and ions) in the chemical system are matched to templates in the force fields. The CTs should also contain the following CT-level properties:

`r_chorus_box_ax`, `r_chorus_box_ay`, ... `r_chorus_box_bx`, ... `r_chorus_box_cz`

that specify the size and shape of the simulation box. The output from the `system_builder` utility meets these conditions.

Residue Matching

Atomic numbers and the bonding pattern (graph isomorphism) are used to match residues to templates. This methodology supports nonstandard atom or residue PDB names without modification. Atom and residue names in a force field need not be edited. In particular, `viparr` will identify the N- and C-terminus versions of the residues correctly, as well as protonated and deprotonated versions of a residue, even if they are not explicitly mentioned as such in the input file. Modification of atom and residue names for clarity is allowed.

Residue and Atom Ordering

The atom ordering in the input file is retained in the output structure file. The residue numbering, which also remains unaltered, can begin with any integer (including negative integers) and does not need to be contiguous (`viparr` constructs a contiguous set of indices that it uses internally). Residues with different chain names can have the same residue number. To aid in diagnosing problems with the input structure file, messages involving residues have the form

<"chain-name", residue-number> (residue-name)

and are usually preceded by a structure number.

Output Format

A compressed force field representation is written when all the residues in a CT are the same. For a CT that only contains water molecules, this means that force field parameters are written only for a single water molecule.

A version number, which is associated with a particular version of `viparr` along with the versions of the built-in force fields and their associated plugins, is written into the output structure file (in the `ffio_version` field). You are responsible for versioning your custom force fields, using Perforce, for instance.

Potential Sources for Errors

If `viparr` reports that it cannot match a residue, please check the following:

- The template for the residue is really in the force field selected.
- Atom numbers for the residue are correct in the input structure file.
- Bonds for the residue are correct in the input structure file.

8.4 Specifying Multiple Force Fields

Multiple force fields can be specified using `viparr`. Common scenarios for this are outlined below.

Different force fields for different parts of a chemical system

In this scenario, one force field is used for one part of a chemical system (e.g., the protein) and another force field for another part (e.g., the water molecules). In this case, each residue in your chemical system matches a template in exactly one of the specified force fields (warning messages are printed otherwise).

Examples:

```
# use spc water model
$SCHRODINGER/run -FROM desmond viparr.py -f amber99 -f spc \
example.mae output.mae
# use tip3p water model
$SCHRODINGER/run -FROM desmond viparr.py -f amber99 -f tip3p \
example.mae output.mae
```

Combining components of two or more force fields

In this case, residues in the chemical system match templates in more than one of the specified force fields (warning messages are printed). All matching force fields are applied. For example, one force field provides the angle parameters for the residues, while another force field provides the dihedral parameters. This can work if the force field components are disjoint and there is no conflict in what parameters are assigned to each component.

Overriding parameters in a force field

Similar to the scenario mentioned above, residues in the chemical system match templates in more than one of the specified force fields (warning messages are printed when this happens) and all matching force fields are applied. However, if two or more force fields provide parameters for the same term (e.g., two force fields provide parameters for the angle between atoms 1, 2, and 3) the conflict is resolved by using the parameters from the first force field listed on the command line that matches the residue.

In all cases, if a bond exists between two residues that are not matched by the same force field, `viparr` exits with an error message. You should correct the problem so that this bond is recognized by one of the selected force fields. The force fields must have consistent van der Waals mixing rules: `viparr` exits with an error message if they do not.

When using multiple force fields, `viparr` does the following:

- If any residue matches more than one template in a force field, `viparr` exits with an error. No `viparr` force field should contain identical templates.
- If any residue name is matched to a force field template with a different name, a message is printed. A maximum of 5 messages are printed per residue-template name pair.
- If there are any unmatched residues, `viparr` prints all unmatched residues and exits with an error. A maximum of 5 messages are printed per unmatched residue name.
- If any residue is matched by more than one of the selected force fields, `viparr` prints a warning message. You should be sure that you intended multiple force fields to match and if so that the appropriate one was selected.

8.5 User-Defined Force Fields

A force field is composed of the following files:

- a template file
- a force-field parameter file, generally for each component of the force field. Angles, proper dihedrals, van der Waals, etc. are examples of force field components

- a set of plugin programs that process the parameter files
- a rules file, which includes a list of plugin programs that `viparr` can call

A set of plugin programs has been provided for the built-in force fields. In most cases, user-defined force fields can use these plugin programs. These plugin programs are located in `VIPARR_PDIR`. `VIPARR_PDIR` is an environment variable that should be set to `$SCHRODINGER/desmond-vversion/lib/Linux-x86/viparr_plugins`.

The other files, namely the templates file, parameter files, and rules file that specify a given force field are placed in a force field directory. The force field directories for the built-in force fields are located in the directory given by the environment variable, `VIPARR_FFDIR`, which should be set to `$SCHRODINGER/desmond-vversion/data/viparr`.

The `-d`, `-m` and `-p` options, described in [Table 8.1](#), are provided for working with user-defined force fields.

8.6 Known Issues

Two or more geometrically identical hydrogen atoms (such as in CH2 or CH3) are treated identically. If the force field needs to treat them differently, you must ensure that the residue name exactly matches the force field template name.

When you import a structure into Maestro, make sure you correct any problems that Maestro detects, especially those involving the atomic numbers of ions.

Utilities

The Desmond distribution contains a number of utilities for performing a range of specific tasks, apart from those described previously. These utilities are described in this chapter.

9.1 solvate_pocket

The `solvate_pocket` utility is a tool for solvating subregions of a system with water, and in particular to solvate buried regions in a protein or protein-ligand complex. To solvate such regions in a thermodynamically consistent manner, `solvate_pocket` uses a grand canonical Monte Carlo approach to sample both the water molecule positions and the number of water molecules present, by attempting to introduce or remove water molecules in a Metropolis-like manner.

9.1.1 Methodology

In grand canonical methods, a value of the chemical potential is set and the simulation samples the number of water molecules in a manner consistent with the chemical potential and the specified temperature. In principle, even the water in buried pockets is in equilibrium with bulk water and so one should conduct the simulation using the excess chemical potential for bulk water (for TIP4P this is about -6.95 kcal/mol).

The `solvate_pocket` utility samples the number and conformation of water molecules within an orthorhombic region of the simulation cell using grand canonical Monte Carlo (GCMC) in the μ VT (constant chemical potential, volume and temperature) ensemble. As a short-cut it does not use periodic boundary conditions other than to center the orthorhombic cell in the simulation prior to simulating water molecules in the sampled region. As such, the subregion sampled by `solvate_pocket` should be surrounded overall by bulk solution even if the sampled region itself is not fully solvated. In its current form `solvate_pocket` expects a CMS file as input and produces a CMS file containing the final conformation of the system from the GCMC simulation.

Interactions are calculated using the real-space part of the Ewald sum only, an approximation that still gives reasonable energetics. For instance, the chemical potential of bulk TIP4P water using this approach is around -7.17 kcal/mol.

The `solvate_pocket` utility uses the concept of a “pass”, which is roughly 1 translation/rotation move for each molecule being sampled. This can be useful because 1 pass is very roughly

equivalent to a MD time step in terms of the amount of sampling for small molecules like water. The passes can include insertion and deletion moves as well. Both types of moves are required in order to equilibrate the number of water molecules in the system.

9.1.2 Command Syntax

The syntax of the `solvate_pocket` command is as follows:

```
$SCHRODINGER/utilities/solvate_pocket [options] [-NJOBS subjobs]
      -spd command-file -icms input-cms-file -ocms output-cms-file
```

The options are given in [Table 9.1](#).

Table 9.1. Options for the `solvate_pocket` command.

Option	Description
-ChargeA <i>n</i>	For FEP calculations, specify the lambda value for the charges for the original molecule.
-ChargeB <i>n</i>	For FEP calculations, specify the lambda value for the charges for the mutated molecule.
-h[elp] -HELP	Print the usage message.
-lct <i>N</i>	Use the <i>N</i> th structure in the input CMS file to define the region sampled by <code>solvate_pocket</code> . <i>N</i> must be greater than 1 (the full system structure). Multiple instances of this option can be used to specify more than one structure for the region definition. Used to override the region specified in the command file. Not compatible with -lmae.
-lmae <i>ligand-file</i>	Specify the name of the ligand file used to define the region sampled by <code>solvate_pocket</code> . Used to override the region specified in the command file. Not compatible with -lct.
-seed <i>number</i>	Provide a seed (an integer) for the random number generator.
-vdwA <i>n</i>	For FEP calculations, specify the lambda value for the van der Waals potentials for the original molecule.
-vdwB <i>n</i>	For FEP calculations, specify the lambda value for the van der Waals potentials for the mutated molecule.
-NJOBS <i>jobs</i>	Run <code>solvate_pocket</code> the specified number of times, with the same input but different random seeds. The output CMS file is the file with the number of waters that is closest to the average over all the subjobs.

Here, *command-file* is the name of the command file for `solvate_pocket`, which has the extension `.spd`, and is described in [Section 9.1.3](#); *input-cms-file* and *output-cms-file* are the input and output CMS (composite model system) files and usually have a `.cms` extension. The optional *ligand-file* is a file containing the ligand. If it is present, it is used to define the subregion for solvation. By default, the maximum and minimum *x*, *y*, and *z* values in the command file are used. If any of `-ChargeA`, `-ChargeB`, `-vdwA`, or `-vdwB` is used, all of them must be used and it is assumed that this run is for a relative FEP calculation.

The `solvate_pocket` utility is run under Job Control by default.

Sampling can be improved by running multiple instances of `solvate_pocket` with the same input file but different random seeds. These jobs can be run simultaneously by specifying more than one processor with the `-HOST` option. The number of instances can be specified with the `-NJOBS` option; by default it is set to the number of processors requested. When all subjobs are complete, the output CMS file with the number of water molecules that is closest to the average number across the subjobs is returned as the output structure file for the job as a whole.

A `solvate_pocket` run on a binding site that can contain around 15 water molecules (about 450 Å³) can take between 10 minutes and an hour. The cost of the simulation increases approximately as the square of the number of water molecules sampled. It is unlikely that `solvate_pocket` is needed to prepare systems for NPT simulations if there are no buried pockets.

9.1.3 Command File Syntax

The `solvate_pocket` utility uses a *keyword = value* syntax that is at least nominally consistent with Ark syntax. This means that you can use the same keywords in both the `solvate_pocket` command file and in the `solvate_pocket` stage of the `multisim` input file. Comments are lines that begin with a `#` character. Blank lines are ignored. Strings are limited to 80 characters and must not contain a new line character. The keywords are described in [Table 9.2](#).

Table 9.2. Keywords for the `solvate_pocket` utility

Keyword	Description
<code>chargeA</code>	For FEP calculations, the lambda value for the Coulombic potentials for the original molecule. Must be used with <code>chargeB</code> , <code>vdwA</code> , <code>vdwB</code> .
<code>chargeB</code>	For FEP calculations, the lambda value for the Coulombic potentials for the mutated molecule. Must be used with <code>chargeA</code> , <code>vdwA</code> , <code>vdwB</code> .
<code>chemical_potential</code>	The chemical potential of water in kcal/mol. Required. Recommended: -7.17 kcal/mol (for TIP4P).

Table 9.2. Keywords for the *solvate_pocket* utility (Continued)

Keyword	Description
<code>cut_off</code>	The cutoff distance for calculating electrostatic and Lennard-Jones interactions, in angstroms. Required. Recommended: 9.0 Å.
<code>distribution_window</code>	The window used for calculating the distribution of the number of water molecules. If this is larger than the value specified by <code>num_passes</code> it is reduced to that value. If early termination upon convergence is used and the number of passes carried out is smaller than specified by <code>distribution_window</code> , that number of passes is used instead.
<code>init_num_passes</code>	The number of Monte Carlo passes to perform in the equilibration prior to the production Monte Carlo run. Recommended: 10000.
<code>lig_ct_nums</code>	List of structures in the input CMS file to define the region sampled. The value must be greater than 1, which is the full system CT.
<code>max_dtheta</code>	The maximum change used for the cosine of theta (Euler angle) for a combined translation/rotation move, in radians. Required. Recommended: 0.0654.
<code>max_disp</code>	The maximum change used in each of the <i>x</i> , <i>y</i> , and <i>z</i> directions for a combined translation/rotation move. Required. Recommended: 0.105 Å.
<code>max_dpsi</code>	The maximum change used for phi and psi (Euler angles) in radians for a combined translation/rotation move. Required. Recommended: 0.318.
<code>name</code>	A descriptive name for the simulation.
<code>num_delete</code>	The number of attempts to delete a single water molecule per pass. Required. Recommended: half the number of water molecules expected.
<code>num_insert</code>	The number of attempts to insert a single water molecule per pass. Required. Recommended: half the number of water molecules expected.
<code>num_passes</code>	The number of Monte Carlo passes to perform in the production Monte Carlo run. Required. Recommended: at least 20000.
<code>num_trans_rot</code>	The number of water molecule translations to attempt per pass. Required. Recommended: approximately the number of water molecules expected to reside in the region being sampled.
<code>output_structure</code>	Controls the output structure produced. Valid values are: <i>final</i> —The last configuration sampled. This is the default. <i>average</i> —The last structure from the trajectory that has a number of dynamic water molecules closest to the average. <i>most_frequent</i> —The last structure from the trajectory that has a number of dynamic water molecules closest to the most visited number. If <i>average</i> or <i>most_frequent</i> are given then both <i>trajectory_freq</i> and <i>distribution_window</i> must be given.

Table 9.2. Keywords for the `solvate_pocket` utility (Continued)

Keyword	Description
<code>pass_term_window</code>	The number of passes over which the slope of the standard deviation of the number of water molecules is calculated. This keyword may be used to terminate the calculations before the number of passes specified by <code>num_passes</code> has been completed. The simulation portions will run for at least twice this duration before early termination occurs. Default: 100000.
<code>renum_waters</code>	If a nonzero value is given, renumber the water residues, starting from 1. Otherwise retain the input water residue numbering.
<code>sample_xmin</code> <code>sample_xmax</code> <code>sample_ymin</code> <code>sample_ymax</code> <code>sample_zmin</code> <code>sample_zmax</code>	Limits of the orthorhombic box in which the water molecules will be sampled. For WaterMap jobs this box should enclose the binding site. This box should be completely enclosed by atoms after it is centered in the input periodic cell.
<code>short_dist</code>	The closest acceptable approach for any two atoms, in angstroms. Required. Recommended: 1.0 Å.
<code>temperature</code>	The temperature used in the Monte Carlo simulation. Required.
<code>term_std_slope</code>	Threshold for the standard deviation of the number of water molecules. The calculation is terminated if the standard deviation falls below this value. Default: 0.00001.
<code>trajectory_freq</code>	If present, specifies the frequency at which structures are saved to a Maestro pose viewer file (<code>_pv.mae</code>). The first structure contains the fixed portion of the system. Structures containing only the water molecules are written every n passes, where n is the value assigned to <code>trajectory_freq</code> .
<code>update_frequency</code>	The number of passes between updates in the log file. Default: 10.
<code>vdwA</code>	For FEP calculations, the lambda value for the van der Waals potentials for the original molecule. Must be used with <code>vdwB</code> , <code>chargeA</code> , <code>chargeB</code> .
<code>vdwB</code>	For FEP calculations, the lambda value for the van der Waals potentials for the mutated molecule. Must be used with <code>vdwA</code> , <code>chargeA</code> , <code>chargeB</code> .

Below is an example `solvate_pocket` command file.

```
name= solvate_pocket example command file

temperature= 298.15
init_num_passes= 10000
num_passes= 100000
```

```

update_frequency = 10
pass_term_window = 10000
term_std_slope = 0.00001

num_trans_rot= 20
num_delete= 5
num_insert= 5

max_disp= 0.105
max_dpsi= 0.318
max_dctheta= 0.0654

sample_xmin= -8.0
sample_xmax= 8.0
sample_ymin= -8.0
sample_ymax= 8.0
sample_zmin= -8.0
sample_zmax= 8.0

cut_off= 9.0
short_dist= 1.0
chemical_potential= -7.17

```

This file instructs `solvate_pocket` to sample water within a cube, 16 Å on a side, centered at 0.0. 10,000 passes will be used to equilibrate the system before the production simulation starts. The production simulation will run up to 100,000 passes but will terminate before that number is reached if the slope of the standard deviation for the number of molecules drops below 0.00001 over a window of 10,000 passes.

9.2 manipulate_trj.py

This script can be used to generate a new trajectory from a list of input trajectories. The trajectory includes the `.cms` and `.idx` files that are needed to display the trajectory in Maestro. The command syntax is as follows:

```

$SCHRODINGER/run -FROM desmond manipulate_trj.py [-h|--help]
  [--mode={merge|concat [--dt=time]}] [-icms=cmsfile] output-trj
  input-trj1 [input-trj2 ... ]

```

This script currently supports two modes of operation, specified by the `--mode` option:

- **merge**—Multiple input trajectories are merged into one new trajectory based upon the chemical time. For example, if the trajectories $A = [a_0, \dots, a_n]$ and $B = [b_0, \dots, b_n]$ are merged, all frames from trajectory A whose chemical time is larger than that for b_0 are

discarded. Here, the trajectories are represented as a list of frames a_i and b_i . This is the default mode, and is useful for merging trajectories that are continued in a new run.

- `concat`—Frames from the input trajectories are simply concatenated, and the time for each frame is reset to account for the new ordering. You can specify the time between two adjacent frames in ps with the `--dt` option.

You can select a subset of the frames present in each input trajectory with a syntax similar to that used for Python lists. If a list is used, the entire trajectory specification must be quoted. The examples below illustrate the syntax:

<code>in_trj</code>	include all frames from <code>in_trj</code>
<code>"in_trj[:]"</code>	include all frames from <code>in_trj</code>
<code>"in_trj[0,6,8,10]"</code>	include frames 0, 6, 8 and 10 from <code>in_trj</code>
<code>"in_trj[1:3:, 5]"</code>	include frames 1, 2 and 5 from <code>in_trj</code>
<code>"in_trj[0, 4:11:2, 20]"</code>	include frames 0, 4, 6, 8, 10 (4–11 in increments of 2), and 20 from <code>in_trj</code>

The frame index is sorted in ascending order for each trajectory before any subset is selected. The extension used in the input CMS file is preserved in the output CMS file (e.g. `-out.cms`).

Some examples are given below.

- To merge `in1_trj` and `in2_trj`:

```
$SCHRODINGER/run -FROM desmond manipulate_trj.py out_trj in1_trj in2_trj
```
- To concatenate frame 0, 3, 6, 13, 5 of `in1_trj` and all frames of `in2_trj`:

```
$SCHRODINGER/run -FROM desmond manipulate_trj.py --mode concat out_trj "in1_trj[0:7:3, 13, 5]" "in2_trj[:]"
```

9.3 amber_prm2cms.py

The utility `amber_prm2cms.py` can be used to convert Amber input files (a `prmtop` file plus a `prmcrd` file) into a Desmond `.cms` file. The syntax for this command is:

```
$SCHRODINGER/run amber_prm2cms.py -c prmcrd -p prmtop -o cms-file
```

where *prmcrd* is the Amber `prmcrd` coordinate or restart file, *prmtop* is the Amber parameter and topology file, and *cms-file* is the output CMS file.

The Amber parameter and topology file can be generated by the `PARM` or `LeaP` programs in the Amber package. As well, `AnteChamber`, which is available for free under a GNU general

public license from <http://ambermd.org/antechamber/antechamber.html> can also be used to construct Amber input files.

9.4 mold_gpcr_membrane.py

The utility `mold_gpcr_membrane.py` can be used to embed a GPCR in a membrane.

With more GPCR X-ray structures being determined over the last few years and the inherent flexibility of GPCR structures, there is great interest in simulating them, particularly within a membrane. Properly constructing the system to simulate can be tedious, time consuming and error prone. Proper alignment of the GPCR in the membrane can be difficult to attain. As well, the relaxation of the membrane around the protein can take a very long time.

Most GPCR proteins are simulated based on either b2 or rhodopsin structures. As a result membranes that are equilibrated around b2 or rhodopsin structures are likely to be able to accommodate another GPCR structure with only relatively mild clashes. The utility `mold_gpcr_membrane.py`, can automatically align a GPCR model to a2a, b1ar, b2ar or rhodopsin and use the pre-equilibrated membrane structure from the latter to reduce the membrane relaxation time and the potential structural problems that can arise during membrane equilibration.

To run `mold_gpcr_membrane.py`, use the following command:

```
$SCHRODINGER/run mold_gpcr_membrane.py [options] {-j|-JOBNAME} jobname
-r[reference] reference-structure -l[lipid] lipid-type input-mae-file
```

The input Maestro file contains one or more model GPCR systems as separate structures. An output structure file is generated for each structure present in the input file. The files are named `jobname_n-out.cms`, where *n* is the index of the structure in the input file, starting from 1. If there is only one input GPCR structure then *n* is omitted.

The arguments and options are given in [Table 9.3](#).

Table 9.3. Options for the mold_gpcr_membrane.py script.

Option	Description
<code>-j -JOBNAME jobname</code>	Job name.
<code>-f -reference_file filename</code>	User reference template file. File that contains equilibrated template/membrane systems. Contact Schrödinger if you want to set up your own template file.
<code>-l[lipid] lipid-type</code>	Specify the type of lipid in the membrane. Valid values are dppc or popc.

Table 9.3. Options for the `mold_gpcr_membrane.py` script.

Option	Description
<code>-r[reference]</code> <i>reference</i>	Specifies the GPCR template. Valid values are a2a, b1ar, b2ar, or rho (rhodopsin).
<code>-z[_dist]</code> <i>distance</i>	Buffer distance for solvation of the system. Default: 10 Å.
<code>-h[elp]</code>	Show usage message and exit
<code>-v[ersion]</code>	Show program version and exit

9.5 trajectory_extract_frame.py

The `trajectory_extract_frame.py` script can be used to extract selected frames from a trajectory into a series of output structure files. To run this script, use the following command:

```
$SCHRODINGER/run -FROM desmond trajectory_extract_frame.py [options]
               cmsfile trjdir
```

The options are given in Table 9.4. The input CMS file and trajectory directory must be specified; these have related names by default: `myjob.cms` and `myjob_trj`. The output files are named with a base name and a sequential index.

Table 9.4. Options for the `trajectory_extract_frame` command.

Option	Description
<code>-h --help</code>	Show usage message and exit.
<code>-a asl </code> <code>--asl=asl</code>	ASL expression to specify which part of the structure is to be extracted. Not valid with <code>-o</code> . Default: <code>all</code> .
<code>-b basename </code> <code>--basename=basename</code>	Base name of the output structure file. The file name is constructed as <code>basename_index.outext</code> , where <i>index</i> is a sequential index (not the frame index). The default is the base name of the input <code>.cms</code> file.
<code>-o outext </code> <code>--output=outext</code>	Format of the output structure file, given as the file extension. Valid values are <code>mae</code> , <code>cms</code> and <code>pdb</code> . Default: <code>mae</code> .
<code>-f frames </code> <code>--frames=frames</code>	List of frames to extract. The list is comma-separated and can include ranges specified as <code>n:m</code> . The default is to extract all frames.

9.6 desmond_restraints.py

This command-line script allows you to add or remove distance, angle and dihedral angle restraints to a CMS file for use in a Desmond simulation. The restraints can be flat-bottomed (i.e. 0 for some range around the equilibrium value) if the width parameter is specified. The syntax of the command is:

```
$SCHRODINGER/run -FROM desmond desmond_restraints.py [options]
-c restraint.cfg input-cms-file output-cmsfile
```

The options are given in [Table 9.5](#).

Table 9.5. Options for the desmond_restraints.py script.

Option	Description
-v[ersion]	Show the program version and exit.
-h[elp]	Show the usage message and exit.
-c <i>input-file</i>	Specify the input config file that contains the restraint definitions.
-r <i>rcut</i>	Cutoff distance for constraints. To correctly compute angle and improper restraint interactions, each group of restrained particles should fit within a sphere of a given radius, which can be set with this option. If they do not, Desmond halts with an error. Default: 9.0 Å.
-d	delete restraints

Example of adding restraints:

```
desmond_restraints.py -c restraint.cfg input_cms_file out_cms_file
```

Example of deleting restraints:

```
desmond_restraints.py -c restraint.cfg -d input_cms_file out_cms_file
```

The syntax of a restraint in the restraint config file is as follows:

```
[ ct atoms k s r0 ]
```

Here, *ct* is the index of the structure (CT) in the Maestro input file, *atoms* is 2, 3 or 4 atom indices in the CT (2 for a distance, 3 for an angle, 4 for a dihedral), *k* is the force constant in the same units as normal bonded interactions, *s* is the half-width of the flat bottom (optional) and *r0* is the reference value of the coordinate (optional).

Example of a restraint config file:

```
# reference value (r0) is obtained from a current structure when
# reference value is not given.
# sigma value (s) is set to be zero if there is no input value.
ffio_stretch_fbhw = [
# ct atom1 atom2 k [s r0]
[2 1 2 0.1 ] # without reference value
[2 2 3 0.1 0.0 1.0] # with reference value
]
ffio_angle_fbhw = [
# ct atom1 atom2 atom3 k [s r0]
[2 2 3 40 0.1 ] # without reference value
[2 1 3 40 0.1 0.0 120.0] # with reference value
]
ffio_improper_fbhw = [
# ct atom1 atom2 atom3 atom4 k [s r0]
[2 2 3 4 5 0.1 ] # without reference value
[2 1 2 3 4 0.1 0.0 180.0] # with reference value
]
```

9.7 rebuild_cms.py

This utility creates a full model system from either the component CTs (solute, membrane, and solvent) or a full system CT. To generate the full model system with all CTs from a set of components, run the following command:

```
$SCHRODINGER/run -FROM desmond rebuild_cms.py -make_full_ct input.cms
output.cms
```

To generate the full model system from just the full system CT, run the following command:

```
$SCHRODINGER/run -FROM desmond rebuild_cms.py -make_comp_ct
[-solvent_asl ASL] [-membrane_asl ASL] input.mae output.cms
```

The solvent and the membrane are automatically detected, but you can specify them by using ASL expressions. If the box vector is not present, the box size is estimated from the system size.

9.8 Other Scripts

In addition to the utilities described above, which are provided with the distribution, there are some other useful scripts available from the [Script Center](#) on the Schrödinger web site, or in the distribution, listed below. These should be run with `$SCHRODINGER/run`.

- `trajectory_cluster`—cluster frames from a trajectory (Script Center)
- `trajectory_delete_water`—create a new trajectory without any water molecules present in the trajectory provided. (Script Center)
- `predict_remd_temp.py`—suggests temperatures for a replica exchange simulation given a CMS file. (distribution)

You can run this script with the following command (without line breaks):

```
$SCHRODINGER/run $SCHRODINGER/mmshare-vversion/lib/Linux-x86_64/  
lib/python2.7/site-packages/schrodinger/application/desmond/  
predict_remd_temp.py
```

- `assign_custom_charge.py`—assign custom charges defined in a reference file to the structure in a Maestro file. Intended for use with FEP simulations. (distribution)

The multisim Utility

The `multisim` utility is useful for running tasks that consist of a sequence of steps, such as relaxation of a system followed by a production simulation. Maestro runs `multisim` for jobs that include multiple stages. This appendix provides detailed information on how to run `multisim` and the format of `multisim (.msj)` files.

Multisim is used by both Desmond and MCPRO⁺ for running jobs.

A.1 Running multisim

The basic usage information for `multisim`, including examples, is given in [Section 6.2 on page 70](#). This section describes more advanced `multisim` features such as template `multisim` commands, node locking, restarting `multisim` jobs, and obtaining information from `multisim` checkpoint files.

A.1.1 Template multisim Commands

Most `.msj` files produced by Maestro end with a commented out example of the command needed to run the job. For example:

```
$SCHRODINGER/utilities/multisim -JOBNAME example -HOST master-job-host
    -SUBHOST subjob-host -maxjob 0 -cpu "2 2 2" -i example.cms
    -m example.msj -c example.cfg -o example-out.cms
```

To use this command you would replace the following text with the values for your own setup:

- *master-job-host*—the host on which to run the master job (the Python script that manages the `multisim` job).
- *subjob-host*—the host on which to run the subjobs, which are often cpu intensive.
- *example*—the job name and the stem of various file names, which are constructed in a standard way from the job name.
- the cpu specification "2 2 2"—the specification of the CPUs used by the job. This specification can either be a triplet of numbers that defines the spatial decomposition of the system, with the total number of processors being the product of these three numbers; or a single integer, which is the total number of processors, and must have only the factors 2, 3 and 5.

- the value for `-maxjob`, which is used to specify how many subjobs can be queued at the same time. The value 0 means “all subjobs” normally, but is the same as the value 1 for umbrella mode.

A.1.2 Node Locking

Node locking involves requiring all subjobs to run on the same nodes (CPUs) as the master job. Node locking can be turned on by adding `-mode umbrella` to the command line. This feature can be useful on busy computer systems that are controlled by a queuing system, because only the master process is submitted to the queuing system. All subjobs will then run on the nodes allocated to the master process without being resubmitted to the queuing system. When this option is given the `-SUBHOST` option is ignored.

Node locking is supported for FEP jobs. You can run these jobs in a serial queue if you use `-maxjob 1`.

A.1.3 Restarting *multisim* Jobs

The *multisim* job periodically writes out a checkpoint file, which records the current state of the workflow. The checkpoint file is named *jobname-multisim_checkpoint*. This file does not include data produced by subjobs. The checkpoint file is copied back to the job launch directory when the master job stops.

In most cases the *multisim* job can be restarted with a command similar to the following:

```
$SCHRODINGER/utilities/multisim -RESTART myjob-multisim_checkpoint  
-d myjob_stage-out.tgz -JOBNAME myjob
```

where you specify the checkpoint file with the `-RESTART` option and the `-out.tgz` file of the last completed stage with the `-d` option. The `-d` option is usually necessary because the subsequent stages need access to the data produced by the last successfully completed stage. For example, if stage 5 is partially done (some subjobs finished, some did not), then you would also have to include *myjob_4-out.tgz*, and the command to use would be:

```
$SCHRODINGER/utilities/multisim -RESTART myjob-multisim_checkpoint  
-d myjob_5-out.tgz -d myjob_4-out.tgz -JOBNAME myjob
```

This command restarts the job on the same compute hosts as before. Unless the `-HOST` option is given the master job runs on the local host. The subjobs will run using the same number of CPUs as before. If you want to use different hosts to restart a job you can use the `-HOST` and `-SUBHOST` options. The number of CPUs used by the subjobs can also be changed using the `-cpu` option, for example,

```
$SCHRODINGER/utilities/multisim -HOST another_master_host
    -host another_subjob_host -RESTART myjob-multisim_checkpoint
    -d myjob_5-out.tgz -JOBNAME myjob -cpu 1
```

You can modify the characteristics of some stages when you restart a job, by using the `-set` option. For example, if you use the command:

```
$SCHRODINGER/utilities/multisim -RESTART myjob-multisim_checkpoint
    -d myjob_4-out.tgz -JOBNAME myjob
    -set "stage[5].time = 1200.0 stage[5].cutoff_radius = 16.0"
```

then stage 5 is modified so that its subjobs run with the time set to 1200.0, and the `cutoff_radius` set to 16.0. However, if you also provide the stage 5 `.tgz` file and some subjobs from stage 5 have finished, only the unfinished jobs will run with the new settings.

You can restart a job with a different `.msj` file. For instance, the command:

```
$SCHRODINGER/utilities/multisim -RESTART myjob-multisim_checkpoint
    -d myjob_5-out.tgz -JOBNAME myjob -m newworkflow.msj
```

runs the uncompleted stages according to `newworkflow.msj`. This new `.msj` file must contain the same completed stages as the original job. Any of the remaining stages can be modified or deleted, and new stages may be inserted.

To restart a job from a completed stage, you can specify the stage number right after the checkpoint file name. For instance, the command:

```
$SCHRODINGER/utilities/multisim -RESTART myjob-multisim_checkpoint:4
    -d myjob_3-out.tgz -JOBNAME myjob
```

reruns the rest of the workflow starting from stage 4. The archive specified with `-d` must be the one for the prior stage (which is stage 3 in this example).

Some other changes that you can or cannot make are:

- You can restart the job with a different `.cfg` file by specifying the new `.cfg` file with the `-c` option.
- You can restart the job with a different maximum number of simultaneously running subjobs by using the `-maxjob` option.
- You cannot change the node-locking mode.

In most cases, *multisim* automatically detects and uses additional input files needed for a job when it is restarted. If the job fails because an existing input file was not detected when restarting a job you can specify that file by using the `-ADD_FILE` option.

A.1.4 Obtaining Information from multisim Checkpoint Files

To obtain information on the contents of a multisim_checkpoint file, you can use the multisim command with the -probe option:

```
$SCHRODINGER/utilities/multisim -probe myjob-multisim_checkpoint
```

This command summarizes the contents of the checkpoint file without submitting a job to continue the calculation. Below is an example of the output from a successful relative solvation free energy calculation:

```
Probing checkpoint file: biphenyl_to_benzene-multisim_checkpoint
  multisim version: 3.8.3.14
  mmshare version: 19104
    Jobname: biphenyl_to_benzene
    Username: me
  Master job host: rosaleen
    Subjob host: myhost
  CPUs per subjob: "1"
  Original start time: Sat May 1 09:14:46 2010
  Checkpoint time: Sat May 1 19:55:35 2010
    Master job ID: rosaleen-0-4bdc2939
  Structure input file: biphenyl_to_benzene_lig.mae
  Original *.msj file: biphenyl_to_benzene_solvent.msj
```

Stages:

```
  Stage 1 completed.
  Stage 2 completed.
  Stage 3 completed.
  Stage 4 completed.
  Stage 5 was skipped.
  Stage 6 completed.
  Stage 7 completed.
  Stage 8 completed.
```

Current version of multisim is 3.8.3.15

This checkpoint file can be restarted with the current version of multisim.

Note the multisim version: checkpoint files generated by older versions of multisim might not be compatible with the current version. The probe option of the multisim command can be used to detect whether the checkpoint file is supported.

If the job had failed in Stage 5 you would see something like:

```
...
  Stage 5 partially completed. 1 subjobs failed, 11 subjobs done.
    Jobname of failed subjobs:
      biphenyl-to-benzene_5_lambda2
  Stage 6 not run.
```

To resume the job as described earlier, this output tells you that you need to provide the `.tgz` file from stage 4, and that you might want to provide the `.tgz` file from the partially completed stage 5. In this case only one subjob failed in stage 5 so rerunning just that subjob might require significantly less computer time than rerunning the entire stage.

A.2 The *multisim* File Syntax

The *multisim* input file (`.msj` file) generally adheres to the Ark format from D. E. Shaw Research. This is the same syntax that is used in Desmond configuration (`.cfg`) files, which are described in brief in [Appendix B](#) and in full in the *Desmond User's Guide*. Ark format is completely compatible with the syntax used in *multisim* files in Schrödinger Suite 2008, but supports other features.

The syntax rules of the Ark format can be summarized as follows.

- Values are assigned to a keyword with the *keyword* = *value* syntax.
- Values can be numbers, strings, lists, or blocks. Numbers can be integers or real numbers. Strings do not need to be enclosed in quotes unless they contain embedded blanks or span multiple lines.
- A block is one or more settings within a pair of braces, `{ }`. For example: `a = { b = 3 c = 4 }`. The block in this example contains two keyword-value assignments.
- A list is a sequence values that is enclosed by a pair of square brackets `[]`. Elements of a list do not have to be of the same type. An example is `[1 { a = 1 } 5 [2 3]]`.
- Hierarchical expressions involving blocks and lists can be created: there is no limit on nesting of lists and blocks.
- Individual elements of a hierarchy can be set by joining the names of the parents with periods to form a compound key. For example, `a.b.c = 3` sets element `c` of block `b` in block `a` to 3.
- If a keyword is assigned a value twice, the second takes precedence. If the value is a block the blocks are merged and keywords present in each block are assigned the values from the latter block.
- The `=` sign can be omitted if the value is a block value. This means that `a = { b = 1 }` and `a { b = 1 }` are equivalent. This syntax is used to specify the *multisim* stages.

Multisim processing deviates from the Ark standard in the following ways:

- *Multisim* stages with the same name remain separate, otherwise stages that appear more than once in an `.msj` file (such as `simulate` or `minimize`) would be combined into one stage.

The multisim input file consists of a sequence of stages, each of which specifies a particular calculation to be run. A stage begins with a label identifying the type of stage followed by braces enclosing parameters for that stage. A msj file will in general look something like:

```
#an outline for a msj file.
first_stage_name {
    parameter1 = 3000.0
    parameter2 = "this is a string"
    parameter3 = [ "list element 1" 2 "list element 3" ]
}
second_stage_name{
    parameter5 = 20
}
```

The types of stages that are supported for Desmond include the following:

- task—describes the type of job
- minimize—minimize the system
- simulate—run an MD simulation on the system
- system_builder—run the system builder
- replica_exchange—run a replica exchange MD simulation on the system
- lambda_hopping—run a lambda hopping simulation on the system
- solvate_pocket—add or remove water molecules in buried pockets
- analysis—analyze MD simulation trajectories
- fep_analysis—analyze FEP calculations
- extern—custom Python stage
- trim—remove unwanted stage data files
- stop—stop the workflow at this stage

The keywords supported for each of these stages and their default values are described in the sections that follow.

A.2.1 General Keywords

The keywords that can be used in any stage of the multisim job are listed in [Table A.1](#).

Table A.1. General keywords that can be used in any stage.

Keyword	Description
compress	File name pattern of the stage data file. If it is set to an empty string, then the data of this stage is not packaged and compressed. Default: <code>\$JOBNAME_\$STAGENO-out.tgz</code> .
dir	Pattern for the names of the directories used by subjobs of this stage. Default: <code>[\$JOBPREFIX/] [\$PREFIX/] \$MASTERJOBNAME_\$STAGENO [_lambda\$LAMBDA]</code>
jlaunch_opt	Options to add to the command to run Desmond. The options are listed in Table 6.1 on page 68 . Example: <code>["-dp"]</code> turns on use of double precision.
jobname	Jobname pattern for subjobs of this stage. Default: <code>\$MASTERJOBNAME_\$STAGENO [_lambda\$LAMBDA]</code>
prefix	Value of the <code>PREFIX</code> macro, which by default is used to specify the sub-directory name of subjobs (see the <code>dir</code> keyword). Default is an empty string.
should_skip	Skip this stage. Allowed values: <code>true</code> , <code>false</code> . Default: <code>false</code> .
should_sync	Do not start this stage until all subjobs of the previous stage finish successfully. Allowed values: <code>true</code> , <code>false</code> . If it is set to <code>false</code> , this stage is started as soon as any subjob of the previous stage finishes successfully. For FEP jobs, setting this keyword to <code>false</code> can result in earlier completion of the job. Default: <code>true</code> .
struct_output	File name of the final output structure file. This keyword is only effective when set in the last stage, and the setting can be overwritten by the <code>-o</code> option of multisim.
title	The title for the stage. Default: <code>?</code> , which stands for none.

A.2.2 Desmond-Specific Common Keywords

The stages that run Desmond directly (`minimize`, `replica_exchange`, `lambda_hopping`, and `simulate`) accept a common set of keywords in addition to keywords specific to the Desmond task. Keywords that are supported in the front-end config file (see [Appendix B](#)) can be used directly in the corresponding stage.

A.2.3 The restrain Keyword

The `restrain` keyword specifies the atom sets to be restrained. Restraints set in all stages except the `system_builder` stage apply only to the current stage. Restraints set in the `system_builder` stage are “permanent”—they are inherited by all subsequent stages. Permanent restraints can be overridden in a particular stage, but only to increase the restraint force constants. They will still be applied in subsequent stages.

The `restrain` keyword supports the specific values listed in [Table A.2](#), and also supports a block syntax and a list syntax.

Table A.2. Specific allowed values of the `restrain` keyword.

Value	Description
<code>heavy_atom</code>	restrain all heavy (non-hydrogen) atoms
<code>none</code>	Remove the temporary restraints (those set in the previous <code>simulate</code> or <code>minimize</code> stage), but do not change permanent restraints such as those set in the <code>system_builder</code> stage or present in the original <code>.cms</code> file given to the <code>multisim.job</code> . If you use this value in a <code>system_builder</code> stage, it removes all restraints. This is the default.
<code>retain</code>	Keep the restraints as set in the previous stage.
<code>solute</code>	Restrain all solute atoms
<code>solute_heavy_atom</code>	Restrain all heavy atoms in the solute
<code>solvent</code>	Restrain all solvent atoms
<code>solvent_heavy_atom</code>	Restrain all heavy atoms in the solvent

Restraints can be specified using blocks, and ASL expressions can be used to define the atoms that are restrained within restraint blocks. The syntax is:

```
restrain = {atom = atoms fc|force_constant = value
            ref|reference_position = retain|reset|refvalue sigma = sigmavalue}
```

where *atoms* can be `solute`, `solvent`, `solute_heavy_atom`, or `heavy_atom` as before, an atom list, or an ASL expression prefixed by `asl:`. For example,

```
restrain = {atom = asl:all force_constant = 10}
```

restrains all atoms with a force constant of 10 kcal mol⁻¹ Å⁻². Restraints can be applied with respect to the positions of the atoms at the start of the current stage by setting `reference_position` to `reset` or the positions from the previous stage by setting `reference_position` to `retain`. The default is `reference_position=reset`.

Relative restraints can be included by setting *atoms* and *refvalue* appropriately. For distance and NOE restraints, *atoms* should specify exactly 2 atoms; for angle restraints, atoms should specify exactly 3 atoms; and for dihedral restraints, *atoms* should specify exactly 4 atoms. All atoms in the relative restraint must be in the same CT. For NOE constraints, *refvalue* should be a list with two elements, the upper distance then the lower distance; for all other types *refvalue* is the desired distance or angle value, in angstroms or degrees.

Flat-bottomed restraints can be included by setting *sigmavalue* to a distance or an angle tolerance, as appropriate. The restraint potential is set to zero in the range $\text{refvalue} \pm \text{sigmavalue}$. For NOE restraints, *sigmavalue* can be a list with two elements, the width of the flat bottom, and the coefficient of the linear term in the restraint potential at large distances.

Different groups of atoms in the system can be restrained with different force constants by listing the restraint blocks for each set of atoms within a list using the syntax:

```
restrain = [{atom = "asl:ASL1" force_constant = value1}
            {atom = "asl:ASL2" force_constant = value2} ... ]
```

If the ASL expression contains quotes, they must be escaped, either by using a backslash, or by using double quotes in the ASL expression and surrounding the entire expression in single quotes. You can include anisotropic constraints by assigning the force constant in any of these expressions as follows:

```
force_constant = [x-value y-value z-value]
```

A.2.4 The atom_group Keyword

The *atom_group* keyword can be used to define atoms groups within the .cms file. Atom groups can be frozen, restrained or associated with particular thermostats. The atom group is defined by the atom-level property *i_ffio_grp_name*. This keyword can be set to the following values:

- none—Remove all atom groups.
- retain—Keep all atom groups from the previous stage.
- { atom = *atoms* index = *i* name = *name* }—atom group block. Put the specified atoms in the atom group *i* that has the name *name*. The atom keyword accepts the same values for *atoms* as the atom keyword for *restrain*, described above. The index *i* is the value for the *i_ffio_grp_name* property. Desmond only supports index numbers from 0 to 7.
- [*group1* *group2* ...]—Specify multiple atom groups. Each group can be specified as a block, in the above format.

An atom group set in the `system_builder` stage is persistent, which means that it remains defined in all subsequent stages, until the next `system_builder` stage.

A.2.5 The task Stage

The `.msj` file should start with a task stage to specify the type of job. Although Multisim can determine the type of job based on the input structure file provided, this can lead to unpredictable behavior if the file was previously used by a different type of job. Explicitly stating the job type avoids this problem, and also permits Multisim to ensure that the input structure file provided is appropriate for the type of calculation requested.

The task stage has two keywords. The first is `task`, whose allowed values are listed in [Table A.3](#).

Table A.3. Values of the task keyword.

Value	Description
<code>desmond:regular</code>	Non-FEP Desmond job
<code>desmond:fep</code>	Desmond absolute or relative free energy (FEP) job
<code>desmond:afep</code>	Desmond absolute free energy (FEP) job
<code>desmond:auto</code>	Desmond job whose type is determined from the input structure file
<code>mcpro:auto</code>	MCPRO ⁺ job whose type is determined from the input structure file
<code>mcpro:fep</code>	MCPRO ⁺ FEP job

The second is `set_family` which can be used to sets parameters for a family of stages. For example:

```
task {
  task = "desmond:fep"
  set_family = {
    desmond = {
      checkpoint.wall_interval    = 7200.0
      checkpoint.write_last_step = no
    }
  }
}
```

The family of stages here is `desmond`. This family includes `minimize`, `simulate`, `replica_exchange`, and `lambda_hopping` stages. The settings in the `desmond = {...}` block are effective for all subsequent stages belonging to the `desmond` family. If there is a

conflict between the settings in the `set_family` block and the explicit settings in a stage, the latter take precedence. For example:

```
task {
  task = "desmond:fep"
  set_family = {
    desmond = {
      checkpoint.wall_interval = 7200.0
      checkpoint.write_last_step = no
    }
  }
}
simulate {
  checkpoint.wall_interval = inf
}
```

The value of the `checkpoint.wall_interval` parameter is set to `inf`, overriding the setting `checkpoint.wall_interval = 7200.0` within the `set_family` block, because explicit settings in the stage have higher precedence.

Valid stage family names include `desmond` and all stage names. If a stage name is used, the family consists of all stages of that name. For example:

```
task {
  task = "desmond:fep"
  set_family = {
    simulate = {
      checkpoint.wall_interval = 7200.0
      checkpoint.write_last_step = no
    }
  }
}
```

Here all subsequent `simulate` stages will be affected, but `minimize`, `replica_exchange`, and `lambda_hopping` stages will not.

A.2.6 The system_builder Stage

Keywords for the `system_builder` stage are listed in Table A.4. The keywords `restrain` (Section A.2.3 on page 112) and `atom_group` (Section A.2.4 on page 113) can also be used in this stage. When they are used, the settings are persistent: they apply to all subsequent stages unless explicitly overridden by another `system_builder` stage.

Table A.4. Keywords for the `system_builder` stage.

Keyword	Description
<code>assign_forcefield</code>	Reassign the force field. If set to false, the <code>system_builder</code> stage can be used to only rebuild the permanent restraints. Default: true.
<code>box_shape</code>	Specifies the box shape. Allowed values are cubic, orthorhombic, triclinic. Default: cubic.
<code>buffer_width</code>	Minimum distance between the box edge and solute in Angstroms. Default: 10.0 Å.
<code>csb_file</code>	Input composite system builder (.csb) file. This option is not required and is usually used to customize system building beyond the keywords supported by multisim.
<code>distil_solute</code>	Move water molecules from the solute entry (CT) to a separate entry (CT). Allowed values are true, false. For non-aqueous solvents, use 'asl:ASL-expression' to specify the solvent molecule. Default: true.
<code>ion_awaydistance</code>	Minimum distance between the added ions and the atoms that the ions should not be placed near, specified by <code>ion_awayfrom</code> . Valid values are non-negative real numbers. Default: 10.0
<code>ion_awayfrom</code>	Do not place ions close to the specified atoms. Valid value is a list of atom indices. Default: no atoms.
<code>minimize_volume</code>	Reorient the solute to minimize the volume of the simulation box when adding solvent molecules. Default: false.
<code>neutralize_system</code>	Add counter ions to neutralize the system. Allowed values are true, false. Default: true.
<code>rezero_system</code>	Reset the origin of the coordinates to the center of mass of the solutes. Allowed values are true, false. Default: true.
<code>solvate_system</code>	Solvate the system. Allowed values are true, false. Default: true.
<code>solvent</code>	Solvent type. Allowed values are SPC, TIP3P, TIP4P, and TIP4PEW. Default: SPC.

A.2.7 The build_geometry Stage

This stage can be used to build the geometry of a simulation system from its components. It provides a greater degree of flexibility in building a system than the system_builder stage, but it only operates on the geometry, not the force field. All keywords for the system_builder stage are accepted except for assign_forcefield, and the solvent keyword has more values, listed below. The keywords specific to this stage are given in [Table A.5](#).

Table A.5. Keywords for the build_geometry stage.

Keyword	Description
only_merge_ct	Merge the specified CTs into a single CT, and ignore all other settings for the stage. This is useful for building a full system CT from the separate CTs. Valid value is a list of CT indices. Default: [].
preserve_box	When building the geometry of the system, don't change the box size if box information exists in the given solute CT. Valid values are true and false. Default: false.
preserve_ffio	When building the geometry of the system, do not delete the existing ffio_ff blocks in the original CT. Valid values are true and false. Default: true.
rebuild_cms	Extract the component CTs from a full system CT. All other settings are ignored. Valid values are true, false, or a block like: <pre>{ membrane = ASL-expression " " solvent = ASL-expression " " }</pre> If the value is not false, the component CTs are extracted from the full system CT. The block value allows you to specify which atoms belong to the membrane component CT and which belong to the solvent component CT. Default: false.
solvent	Solvent type. Allowed values are water, SPC, TIP3P, TIP4P, TIP4PEW, methanol, octanol, DMSO. Values are case sensitive. water is a synonym for SPC. Default: water.

A.2.8 The assign_forcefield Stage

This stage can be used to assign the force field to a model system. You can either keep or replace existing assignments. The keywords are given in [Table A.6](#).

Table A.6. Keywords for the assign_forcefield stage

Keyword	Description
forcefield	Specify the force field to use. Valid values are: OPLS_2005, CHARMM, AMBER, amber03, amber99, amber94, amber96, amber99SB, amber99SB-ILDN, charmm22nocmap, charmm36_lipids, charmm27, charmm32, oplsa_ions_Jensen_2006, oplsa_impact_2001, oplsa_impact_2005, oplsa_impact_2001, oplsa_impact_2005. These force fields are the ones available from viparr—see Chapter 8 . Values are case-sensitive. Default: OPLS_2005.
water	Specify the water model to use. Valid values are: SPC, SPCE, TIP3P, TIP3P_CHARMM, TIP4P, TIP4PEW, TIP4P2005, TIP5P, none. Default: SPC
humble	Do not overwrite the existing ffio_ff block in the input. Valid values are true, false. Default: false.

A.2.9 The simulate, replica_exchange, and lambda_hopping Stages

The keywords that are specific to the simulate, replica_exchange, and lambda_hopping stages are listed in [Table A.7](#). In addition, the keywords as described in [Section B.6 on page 140](#) can be used in the replica_exchange stage. The keywords restrain ([Section A.2.3 on page 112](#)) and atom_group ([Section A.2.4 on page 113](#)) can also be used in this stage. When they are used, the settings are temporary: they apply only to the current stage.

Table A.7. Keywords for the simulate, replica_exchange, and lambda_hopping stages.

Keyword	Description
fep.model_file	List of structure files, each for use with different lambda windows. This keyword should only be used for FEP jobs.
jin_file	List of auxiliary input files for the stage. Usually only needed when custom plugins are used.
jout	List of auxiliary output files for the stage. Usually only needed when custom plugins are used.

The fep.model_file keyword makes it possible to use different input files for different lambda windows. For instance:


```

task {
    task = "desmond:fep"
}

simulate {
    fep.model_file = ["file1.cms" "file2.cms"]
}

```

Here, the first two windows use the `file1.cms` and `file2.cms` respectively. The rest of the windows use the default input file.

Metadynamics simulations can be run by including a meta block in the `simulate` stage. See [Section B.7 on page 141](#) for the syntax of this block. However, in the multisim file, you must use an ASL expression for the center of mass of a set of atoms, not a list. For example,

```
atom = ['atom.num 2379 2380 2382 2384 2385' 'atom.num 631 642 651']
```

A.2.10 The minimize Stage

The keywords that are specific to the minimize stage are described in [Section B.5 on page 139](#). The keyword `restrain` ([Section A.2.3 on page 112](#)) can also be used in this stage. When it is used, the settings are temporary: they apply only to the current stage.

A.2.11 The solvate_pocket Stage

The keywords for the `solvate_pocket` stage are listed in [Table A.8](#). The only keywords that can be set directly within the `solvate_pocket` stage are `spd_file`, `ligand_file`, `fep.lambda`, and `backend`. The other keywords in the table should be set in the `backend` block. This stage runs the `solvate_pocket` utility, which is described in [Section 9.1 on page 93](#).

Table A.8. Keywords for the solvate_pocket stage

Keyword	Description
<code>spd_file</code>	The name of the <code>solvate_pocket</code> command file. If omitted, the default settings are used.
<code>spd_overwrite</code>	Block that provides settings in Ark syntax to overwrite specific commands in the command file. All keywords other than <code>spd_file</code> and <code>ligand_file</code> must be inside this block.
<code>ligand_file</code>	The name of the ligand file used to define the region sampled. If the name is set to an empty string or omitted, the name <code>jobname-ligand.mae</code> is used. A ligand must be used to define the region.

Table A.8. Keywords for the solvate_pocket stage

Keyword	Description
name	A descriptive name for the simulation. Default: name of standard solvate_pocket command file.
temperature	The temperature used in the Monte Carlo simulation. Default: 300 K
init_num_passes	The number of Monte Carlo passes to perform in the equilibration prior to the production Monte Carlo run. Default: 10000.
num_passes	The number of Monte Carlo passes to perform in the production Monte Carlo run. Default: 100000.
update_frequency	The number of passes between updates in the log file. Default: 100.
pass_term_window	The number of passes over which the slope of the standard deviation of the the number of water molecules is calculated. This keyword may be used to terminate the calculations before the number of passes specified by num_passes has been completed. The simulation portions will run for at least twice this duration before early termination occurs. Default: 100000.
term_std_slope	Threshold for the standard deviation of the number of water molecules. The calculation is terminated if the standard deviation falls below this value. Default: 0.00001.
num_trans_rot	The number of water molecule translations to attempt per pass. This should be set to approximately the number of water molecules expected to reside in the region being sampled. Default: 100.
num_delete	The number of attempts to delete a single water molecule per pass. Default: 25.
num_insert	The number of attempts to insert a single water molecule per pass. Default: 25.
max_disp	The maximum change used in each of the <i>x</i> , <i>y</i> , and <i>z</i> directions for a combined translation/rotation move. Default: 0.105 Å.
max_dpsi	The maximum change used for phi and psi (Euler angles) in radians for a combined translation/rotation move. Default: 0.318.
max_dctheta	The maximum change used for the cosine of theta (Euler angle) for a combined translation/rotation move, in radians. Default: 0.0654.
cut_off	The cutoff distance for calculating electrostatic and Lennard-Jones interactions. Default: 9.0 Å.

Table A.8. Keywords for the solvate_pocket stage

Keyword	Description
short_dist	The closest acceptable approach for any two atoms. Default: 1.0 Å.
chemical_potential	The chemical potential of water in kcal/mol. Default: -7.17 kcal/mol (for TIP4P).
fep.lambda	For FEP calculations, turn on the solvate_pocket calculation and specify the lambda schedule that is in use. This keyword should only be used for an FEP job, and the schedule must be the same as for the FEP calculation itself (as specified in the config file). The value true specifies the default:12 schedule, which is the default for FEP calculations. See Appendix B for more information.

A.2.12 The extern Stage

The extern stage provides an extremely flexible way to include your own Python code in a multisim run. The code can be embedded in the .msj file as a string value assigned to the command keyword. For example:

```
extern {
    command = "
import os;
def main( current_stage, job ):
    os.system( 'ls' )
"
```

In the embedded Python code, you can import and use modules from your Schrödinger Python installation. If you need extra modules, you can pass their file names to multisim by setting the auxiliary_file parameter, and multisim transfers them to the scratch directory of the master job at run time. For example:

```
extern {
    auxiliary_file = [mod1.py mod2.py]
    command = "
import os
import mod1
import mod2
def main( current_stage, job ):
    // does something with mod1 and mod2
    os.system( 'ls' )
"
```

The code given above is run once for each subjob in the previous stage. If you want to run it only once, use `command_once` instead of `command`. For example:

```
extern {
    auxiliary_file = [mod1.py mod2.py]
    command_once = "
import os
import mod1
import mod2
def main( current_stage ):
    // does something with mod1 and mod2
    os.system( 'ls' )
"
```

For scripts that are stage-specific, your code must provide a `main` function that takes two arguments for `command` or one for `command_once`. The first argument in both cases corresponds to information for the current stage, while the second argument for `command` corresponds to information from the previous stage.

For very simple scripts, your code for `command` or `command_once` does not need to provide a `main` function. The following example removes a temporary file if it exists:

```
extern {
    command = "
import os
# Removes a temporary file.
if (os.path.isfile( 'my_temporary_file' )) :
    os.remove( 'my_temporary_file' )
"
```

Without the `main` function, *multisim* cannot pass the current stage and the current job objects to the Python code, but that is presumed to be not needed for simple operations.

The `extern` stage is an advanced feature. Please do not hesitate to contact us for additional information on its use. The keywords for this stage are listed in [Table A.9](#).

Table A.9. Keywords for the extern stage.

Keyword	Description
<code>auxiliary_file</code>	List of files containing extra modules to be transferred to the runtime directory.
<code>command</code>	Command to execute once for each subjob of the previous stage. The command specifies Python code that can span multiple lines.
<code>command_once</code>	Command to execute once for the previous stage. The command specifies Python code that can span multiple lines.

A.2.13 The analysis Stage

Keywords for the analysis stage are listed in [Table A.11](#).

Table A.10. Keywords for the analysis stage.

Keywords	Description
<code>meta.range</code>	For metadynamics simulations, specify the range for each collective variable, in the format [<i>min1</i> <i>max1</i>] [<i>min2</i> <i>max2</i>] ...].
<code>meta.nbin</code>	For metadynamics simulations, specify the number of bins for each collective variable in the analysis.
<code>plotter</code>	Specify the plotter to use to draw images for figures. Allowed values: <code>mplchart</code> , <code>gchart</code> . <code>mplchart</code> generates static .png file for each figure. <code>gchart</code> generates a URL for each figure, and the URLs are saved in the report file. When you open the report file in a browser, the figure images are generated using Google chart APIs. The data for the figures is sent to the Google web site to create the images. Default: <code>mplchart</code> .
<code>prob_profile</code>	Specify how to calculate the probability profiles for the time series given by the <code>time_series</code> parameter. Example: <code>prob_profile = [2 0 360 true]</code> , which means the bin size is 2, the data range is [0, 360), and the range is periodic (<code>true</code>).
<code>report</code>	Specify the name of the report file. Default: <code>\$JOBNAME_report.html</code> .
<code>SEA</code>	Block for specifying the configuration of the simulation event analysis. This block can be the contents of an .st2 file, which can be written from the Simulation Event Analysis panel.
<code>time_series</code>	Specify the variables for which to analyze the time series. Example: <code>time_series = [[dihedral 1 2 3 4] [dihedral 5 6 7 8]]</code> , which specifies two dihedral angles to analyze.

A.2.14 The `fep_analysis` Stage

Keywords for the `fep_analysis` stage are listed in [Table A.11](#).

Table A.11. Keywords for the `fep_analysis` stage.

Keywords	Description
<code>bennett.random_seed</code>	Random seed for the Bennett method. Default: 2111839.
<code>correct_vdw</code>	Calculate the long range dispersion correction for absolute free energy jobs. This keyword has no effect for relative free energy jobs. Default: <code>true</code> .
<code>correct_restr</code>	Calculate the enthalpic correction for position restraints. This parameter does no harm if the production simulations have no restraints. Default: <code>true</code> .
<code>plotter</code>	Specify the plotter to use to draw images for figures. Allowed values: <code>mplchart</code> , <code>gchart</code> . <code>mplchart</code> generates static <code>.png</code> file for each figure. Choosing <code>gchart</code> generates a URL for each figure, and the URLs are saved in the report file. When you open the report file in a browser, the figure images are generated using Google chart APIs. The data for the figures is sent to the Google web site to create the images. Default: <code>mplchart</code> .
<code>report</code>	Specify the name of the report file. Default: <code>\$JOBNAME_report.html</code> .

The free energy can be plotted as different functions of the simulation time. The following settings are added to control these new functionalities:

```
bennett = {
  forward_time = {
    name = "freeenergy_time"
    begin = 100.0
    end   = inf
    dt    = 30.0
  }
  reversed_time = {
    name = "freeenergy_rtime"
    begin = 100.0
    end   = inf
    dt    = 30.0
  }
  sliding_time = {
    name = "freeenergy_stime"
    begin = 100.0
  }
```

```
        end      = inf
        dt       = 30.0
        window   = 500.0
    }
}
```

The `forward_time`, `reversed_time`, and `sliding_time` settings control calculations of the free energy over different ranges of time.

The `forward_time` keyword provides the free energy as a function of the simulation course. In other words, each free energy value of this function is calculated using the trajectory fragment from the time as given by the `begin` setting to the time point. The `dt` keyword indicates the interval between consecutive estimates of the free energy (in this case, 30 ps).

The `reversed_time` keyword functions similarly to the `forward_time` keyword except that the free energy is calculated from each time point to the end of the trajectory.

For the `sliding_time` keyword, the fragment length has a fixed length, specified by the `window` keyword (in this case, 500.0 ps). The beginning of the fragment slides along the trajectory with the step size given by the `dt` keyword (30.0 ps in this example).

A.2.15 The `vrun` and `fep_vrun` Stages

This stage runs `vrun` with the Desmond trajectory from the last `simulate` stage. The `vrun` stage should only be used with non-FEP simulations. For FEP simulations, you should use the `fep_vrun` stage, which has the same syntax. All the parameters from the `simulate` stage can be used with these stages, but the default settings for the following keywords are different:

```
trajectory = false
checkpoint = off
maeff_output = off
eneseq.interval = 0.0
```

Setting `eneseq.interval` to zero means that the energy is computed for every frame of the trajectory.

A.2.16 The `trim` Stage

The purpose of the `trim` stage is to automatically delete unwanted stage data files. A common scenario is that most of the relaxation stage files are not needed if the job is successful. Adding a properly-set `trim` stage at the end of the job automatically removes these files before they are transferred back to the launch directory. The following example illustrates how a `trim` stage might be used:

```
task {  
    task = "desmond:auto"  
}  
minimize {  
    # relaxation stage  
}  
simulate {  
    # relaxation stage  
}  
simulate {  
    # production stage  
}  
trim {  
    save = [-1]  
}
```

The `save` keyword is the only trim-specific keyword and specifies stages for which the output files should be kept. Its value is a list of integers. The integers can be either positive or negative. Positive integers are the stage index, negative integers are relative to the `trim` stage, i.e. they refer to stages counting backwards from the `trim` stage. In this example, `-1` means the first stage before this trim stage, i.e. the production stage or the 4th stage. So the trim stage removes all `.tgz` files that were generated by the earlier stages except for the one generated by the production stage.

The following setting for the above example saves the `.tgz` files generated by the 3rd and 4th stages (the two stages before the trim stage):

```
save = [-1 -2]
```

The following setting saves the `.tgz` files generated by the 2nd, 3rd, and 4th stages

```
save = [2 3 -1]
```

Note that the task counts as a stage.

The Configuration File

The configuration file (hereafter referred to as a config file) describes the nature of the calculation that the Desmond backend should perform. Starting with the 2010 release, there are two styles of config files: a user facing, “front-end” style that is designed to make it easier for users to work with, and a “back-end” style, which provides a very detailed and specific description of the calculation to the back-end itself.

For most Desmond jobs a CMS file and a front-end config file are specified on the command line. Then the software automatically translates the front-end config file into a back-end config file in a manner that depends on, or rather, takes into account, the nature of the system (in the cms file). The appropriate Desmond back-end executable for the job (e.g. `mdsim`, `minimizer`) is determined automatically based upon the contents of the front-end config file. This appendix is mainly focused on describing the syntax of the front-end config file. For information on the back-end config file see the *Desmond User’s Guide*.

The Ark syntax from D. E. Shaw Research is used for both the back-end and front-end config files. The definitive description of the Ark format is available in the *Desmond User’s Guide*. This format is also used by multisim (`.msj`) files. Below is a brief recap of the Ark syntax, followed by detailed documentation of the parameters in the front-end config file.

B.1 Ark Format Syntax Summary

Ark uses a free format syntax, in that is there are no indentation or alignment requirements. Spaces (including some control characters, e.g., tab, return, new line.) are useful only for separating keys and values. The Ark format syntax is case-sensitive, so keys and values with different capitalization patterns are treated as distinct.

At the most basic level the syntax is the familiar key-value pair syntax of the form:

key = *value*

where *key* must be a non-empty string, and *value* must be one of the following three objects: atom, list, and map. A value of the atom type is one of the following: integer, floating number, string, boolean (`true`, `false`, `yes`, `no`, `on`, `off` are recognized as boolean values), or none (`?` is recognized as none). A value of the list of type is of the form: `[value1 value2 ...]`, in other words a list of one or more values wrapped within brackets `[]`, and the elements *value1*, *value2*, ... are values of any type and are separated by spaces. Map type values generally look like:

```
{ key1 = value1
  key2 = value2
...
}
```

where the value is wrapped with braces { }, and elements in the map are key-value pairs.

The form describe above is the canonical form of the syntax. Here is an example of this syntax with nested values:

```
a = {
  b = {
    c = [1 2 4]
  }
  d = string_value
}
```

Ark also supports the so-called “pathname” syntax, where nested keys are separated by a period. The pathname equivalent of the canonical example given above is:

```
a.b.c = [1 2 4]
a.d = string_value
```

B.2 Units

The units used in the configuration file are:

- time: ps
- distance: angstroms
- energy: kcal/mol
- pressure: bar
- surface tension: bar angstroms
- temperature: kelvin

Time can be specified as a positive real number or as the string `inf`, meaning infinity, or “never”.

B.3 Macros in String Values

A macro works as a place-holder, and is replaced by a corresponding value after macro expansion is performed (during conversion of the front-end config file into the back-end config file). The name of a macro starts with a \$ character, followed by a sequence of alphanumeric characters. A macro can be expanded only when its value is defined. If its value is undefined, the macro is left in the string unchanged.

For Desmond, the supported macros are listed in [Table B.1](#).

Table B.1. Supported macros

Macro Name	Value defined	Value
\$JOBNAME	Always	The actual name of the job
\$MASTERJOBNAME	Only in Multisim	The actual name of the master job
\$REPLICA	Only for REMD jobs.	The index of the replica. (Index starts from 0.)

Optional string fragments that contain macros can also be defined. An optional fragment is a portion of a string enclosed by `$[` and `]`. Such a fragment, together with the `$[` and `]` characters, is deleted if there are one or more `$` characters within the fragment. If there is no `$` character within the fragment, only the `$[` and `]` are deleted. This operation is performed after macro expansion. To see its effect, consider the following example.

```
name = "$JOBNAME$_replica$REPLICA$.dE"
```

If we have defined the values of `$JOBNAME` and `$REPLICA` to be `myjob` and `0`, respectively, then the actual value for `name` will be `myjob_replica0.dE`. If we do not give a value to `$REPLICA`, macro expansion yields `myjob$_replica$REPLICA$.dE`. Because there is a `$` within the fragment, this fragment is deleted, so that the final value for `name` is `myjob.dE`.

B.4 Common Front-End Parameters

All currently supported parameters are listed and explained below. Each parameter is in a separate subsection.

B.4.1 `fep`

Common usage:

```
fep = {
  lambda    = "default:12"
  i_window  = ?
  output    = {
    name     = "$JOBNAME$_replica$REPLICA$.dE"
    first    = 0.0
    interval = 1.2
  }
}
```

B.4.1.1 `fep.lambda`

This parameter sets the lambda schedule for FEP calculations. You can set it to a string like `default:12`, which means “use the default scheme with 12 windows”. To use 20 windows with the default scheme, set the parameter to `default:20`. Available lambda schedule schemes are: `default`, `quickcharge`, and `superquickcharge`. You can define your own schedule. For instance, for an alchemical free energy calculation, the following syntax can be used:

```
fep.lambda = {  
    bondedA = [1.0 0.9 ...]  
    bondedB = [0.0 0.1 ...]  
    chargeA = [1.0 0.75 ...]  
    chargeB = [0.0 0.25 ...]  
    vdWA = [1.0 1.0 ...]  
    vdWB = [0.0 0.0 ...]  
}
```

Here, A refers to the original molecule while B refers to the mutated molecule, and the ellipsis represents additional values that you would explicitly include. The number of values (lambda stages) must be consistent in all 6 lists.

Similarly for a total free energy calculation the following syntax may be used:

```
fep.lambda = {  
    coulomb = [0.0 0.1 ...]  
    vdW = [1.0 1.0 ...]  
}
```

B.4.1.2 `fep.i_window`

This parameter is automatically set by `multisim` to the index of the lambda window. You do not normally set or adjust this parameter.

B.4.1.3 Default Lambda Schedules

The default 12 step lambda schedule for relative free energy calculations is:

```
lambda = {  
    bondedA = [1.0 0.916666666667 0.833333333333 0.75 0.666666666667 0.583333333333  
              0.416666666667 0.333333333333 0.25 0.166666666667 0.083333333333 0.0]  
    bondedB = [0.0 0.083333333333 0.166666666667 0.25 0.333333333333 0.416666666667  
              0.583333333333 0.666666666667 0.75 0.833333333333 0.916666666667 1.0]  
    chargeA = [1.0 0.75 0.5 0.25 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0]  
    chargeB = [0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.25 0.5 0.75 1.0]  
    vdWA = [1.0 1.0 1.0 1.0 1.0 0.67 0.46 0.33 0.25 0.19 0.12 0.0]
```

```
vdwB      = [0.0 0.12 0.19 0.25 0.33 0.46 0.67 1.0  1.0  1.0  1.0  1.0]
}
```

The default 12 step lambda schedule for absolute free energy calculations is:

```
lambda = {
  coulomb = [0.0 0.118514957434 0.189778434985 0.24741405481 0.325045439067
0.456296209913 0.674789989504 1.0 1.0 1.0 1.0 1.0 ]
  vdw      = [1.0 1.0 1.0 1.0 1.0 0.674789989504 0.456296209913 0.325045439067
0.24741405481 0.189778434985 0.118514957434 0.0 ]
}
```

B.4.2 cutoff_radius

This parameter sets the cutoff radius for the non-bonded interactions. If the particle mesh Ewald (PME) method is used for electrostatic interactions, this sets the cutoff radius for the real space part of the electrostatic interaction calculations.

```
cutoff_radius = 9.0
```

B.4.3 taper

The taper parameter is used to specify how interactions are truncated at the cutoff. For example,

```
taper = off
```

turns off any special treatment of interactions at the cutoff (i.e. they are just truncated). There are several ways to turn on tapering. The simplest way is to set the `taper` parameter to `on`, and the default, `shift`, tapering scheme will be used. In this scheme the tapering consists of shifting the potential to 0 at the `cutoff_radius`. More elaborate tapering schemes, requiring more detailed information, are supported:

```
taper = {
  method = potential
  width  = 1.0
}
```

where the `method` parameter specifies the type of tapering and has one of the following values: `potential`, `c1switch`, `c2switch`, or `shift`. The `width` parameter controls the range of distances, between $(\text{cutoff_radius} - \text{width})$ and `cutoff_radius`, over which tapering is applied. For more information on tapering methods, see the [Desmond User's Guide](#).

B.4.4 coulomb_method

This parameter sets the method for treating Coulombic interactions in the simulation. For example,

```
coulomb_method = pme
```

requests that the particle mesh Ewald (PME) method be used.

Supported values are: `pme`, `cutoff`.

The computational accuracy of the PME method can be controlled by the so-called Ewald tolerance. The smaller the tolerance is, the more accurate but slower the computation is. The default value of the tolerance is 1E-9, which is very accurate for most simulations. If you want to use a different tolerance value, set the `coulomb_method` parameter explicitly in the following form:

```
coulomb_method = [pme 1E-9]
```

where the first element of the list value must be `pme`, and the second element is the desired tolerance value.

B.4.5 temperature

This parameter sets the temperature of the system. For instance:

```
temperature = 300.0
```

requests that a temperature of 300K be used. For multiple thermostats, you must explicitly set the temperature for each of the thermostats using the following syntax:

```
temperature = [[temp1 group-index1] [temp2 group-index2] ...]
```

where the value of the temperature parameter is a list. Each element of the list is a pair, in which the first value of the pair is the temperature, and the second value is the index of the atom group that should have this temperature.

For simulated annealing jobs, the value of the temperature parameter specifies the temperature schedule of the annealing process in the following form:

```
temperature = [[temp1 time-point1] [temp2 time-point2] ...]
```

where the value of the temperature parameter is a list. Each element of the list is a pair, in which the first value of the pair is the reference temperature, the second value is the time point. Between *time-point1* and *time-point2*, the value of the reference temperature is linearly interpolated from *temp1* to *temp2*, and so on for the remaining points.

B.4.6 annealing

The annealing parameter indicates whether this is a simulated annealing simulation, and takes values of `off` (not a simulated annealing simulation) or `on`. If `annealing` is set to `on` then the temperature parameter should be set to an appropriate temperature schedule.

B.4.7 pressure

The target pressure for simulations involving a barostat can be set as follows:

```
pressure = 1.01325
```

The default coupling scheme of the barostat to the system is isotropic. Use the following format to use another coupling scheme:

```
pressure = [1.01325 anisotropic]
```

where the second element of the list value sets the coupling style. The valid values for coupling style are the following: `isotropic`, `anisotropic`.

B.4.8 surface_tension

This parameter sets the surface tension, e.g.

```
surface_tension = 4E3
```

It is ignored unless the `ensemble` parameter is set to `NPgT` (or the `ensemble.method` parameter is set to `NPgT`).

B.4.9 ensemble

This parameter is used to select the ensemble to use for the simulation. For instance,

```
ensemble = NPT
```

requests that the simulation be done in the constant pressure / constant temperature ensemble with a constant number of particles (atoms).

In the simplest form, you can set the parameter to different ensemble classes, and the default setup for that class is used. Valid values for ensemble class are the following: `NPT`, `NVT`, `NVE`, `NPgT`, `NPAT`, `NPT_Ber`, `NVT_Ber`, `NPT_L`, `NVT_L`. In all of these ensembles the number of particles is constant. `NPgT` represents the ensemble where the pressure, temperature, and surface tension (x-y plane) are also held constant. In the `NPAT` ensemble, in addition to the number of particles, the pressure, temperature, and area of the system (in the xy plane) are held constant.

The ensembles represented by NPT, NVT, NPgT, and NPAT are controlled by a Nosé-Hoover-based algorithm. Those ending with `_Ber` or `_L` use alternative algorithms, namely those based upon the work of Berendsen or Langevin.

The ensemble can be described in more detail using the following form:

```
ensemble = {  
    class = NPT  
    method = MTK  
    thermostat.tau = 1.0  
    barostat.tau = 2.0  
}
```

where the valid values for `class` are: NPT, NPgT, NPAT, NVT, and for `method` are: MTK, NH, Berendsen, Langevin. The `thermostat.tau` and `barostat.tau` parameters set the relaxation times for the thermostat and barostat, respectively.

For the NVE ensemble, the ensemble parameter can be set as:

```
ensemble = NVE
```

or as:

```
ensemble.class = NVE
```

B.4.10 time

The `time` parameter can be used to set the total simulation time. e.g.,

```
time = 1200.0
```

B.4.11 elapsed_time

The `elapsed_time` parameter controls the starting time for the simulation. Normally this is 0. However, if the simulation is a continuation of an earlier one the current time can be set using this parameter, e.g.

```
elapsed_time = 10.0
```

B.4.12 timestep

The `timestep` parameter specifies values for the bonded, near, and far time steps, respectively. e.g.,

```
timestep = [0.002 0.002 0.006]
```


B.4.13 cpu

The `cpu` parameter specifies the total number of processors and, optionally, the domain-decomposition of the system in the x , y , and z directions. For instance,

```
cpu = [1 2 4]
```

specifies that $1 \times 2 \times 4 = 8$ processors should be used and that the system should be decomposed into 1, 2 and 4 domains in the x , y , and z directions. Alternatively, you can set this parameter to a single integer value, which specifies the total number of CPUs, and then the decomposition is automatically done based on the shape of the simulation box for the system. For example:

```
cpu = 8
```

sets the total number of CPUs to 8. The values specified for the domain decomposition must be powers of 2, 3, or 5, or products of these powers.

B.4.14 glue

Glue tries to ensure that the closest images for closely associated molecules are recorded in the output `cms` and trajectory files. This parameter specifies the molecule set. Only one value is supported at present: `solute`. The parameter can be set with:

```
glue = solute
```

B.4.15 trajectory

The `trajectory` parameter is used to control how the trajectory is written.

```
trajectory = {
  name           = "$JOBNAME$_replica$REPLICA$_trj"
  first          = 0.0
  interval       = 4.8
  periodicfix    = true
  write_velocity = true
  frames_per_file = 25
}
```

The `periodicfix` parameter, if set to `true`, instructs Desmond to wrap molecules as a unit within the periodic boundary conditions rather than to wrap individual atoms (potentially splitting molecules across the unit cell in the recorded trajectory). The `first` parameter is used to indicate at what time to start recording the trajectory, while `interval` controls how often to write trajectory frames. The `frames_per_file` parameter is used to control the number of frames written to each file in the trajectory. Large numbers of files can cause very slow copying and reading of the trajectory, and can even cause failure. You should set this parameter

so that the total number of files is small enough for the file system to handle efficiently (no more than a few thousand). Fewer files also reduces the IO overhead when reading the files.

B.4.16 eneseq

The `eneseq` parameter controls when to start recording the `.ene` file (`first`), how often to update it (`interval`), and the precision of the energies (`precision`). The minimum precision is 8 figures.

```
eneseq = {  
    name      = "$JOBNAME$_replica$REPLICA$.ene"  
    first     = 0.0  
    interval  = 1.2  
    precision = 9  
}
```

B.4.17 checkpoint

The `checkpoint` parameter specifies the name of the checkpoint `.cpt` file, when to start recording (`first`) and how often to overwrite it (`interval`).

```
checkpoint = {  
    name      = "$JOBNAME.cpt"  
    first     = 0.0  
    interval  = 240.0  
    write_last_step = yes  
}
```

To let Desmond periodically write out a `.cpt` file at certain wall time interval, you can set `checkpoint` in this form:

```
checkpoint = {  
    name          = "$JOBNAME.cpt"  
    wall_interval = 3600.0  
    write_last_step = yes  
}
```

where `wall_interval` sets the wall time interval in seconds. You can turn off wall time interval by either setting it to `inf` (`wall_interval = inf`) or not including it at all. If the `wall_interval` parameter is set to a finite value, then the `first` and `interval` parameters have no effect.

You can turn off recording the `.cpt` file altogether using:

```
checkpoint = off
```

B.4.18 maeff_output

The `maeff_output` parameter specifies the name of the output `.cms` file, when to start recording this file (`first`), how often to overwrite it (`interval`), and the number of significant figures given in floating-point numbers (`precision`). It also specifies the name of the `.idx` file (used for recording the trajectory name).

```
maeff_output = {
  name       = "$JOBNAME$_replica$REPLICA$-out.cms"
  trjidx     = "$JOBNAME$_replica$REPLICA$-out.idx"
  first      = 0.0
  interval   = 120.0
  precision  = 8
}
```

B.4.19 randomize_velocity

Sometimes it is useful to randomize the velocities of the atoms at the start of or during a simulation. The `randomize_velocity` parameter specifies when to start randomizing velocities (`first`), how often to do it (`interval`) and the random number seed used for generating the velocities (`seed`). The `temperature` parameter sets the target temperature for the randomized velocities.

```
randomize_velocity = {
  first       = 0.0
  interval    = inf
  seed        = 2007
  temperature = '@*.temperature'
}
```

The value `'@*.temperature'` means the value of the `temperature` parameter that is in the parent map of the `randomize_velocity` parameter. The `@` symbol in the value is referencing operator, `*` means the parent map of the current map, so `*.temperature` means the temperature in the parent map. This syntax is a general feature in the sense that every parameter can be set syntactically in this way. For example:

```
randomize_velocity.first = '@*.trajectory.first'
```

sets `randomize_velocity.first` to the value of `trajectory.first`. Circular references (which are possible syntactically) are caught during the initial job-launch processing.

B.4.20 simbox_output

The `simbox` parameter controls how Desmond records information about the simulation box. This parameter specifies the name of the file, when to start recording the file (`first`) and how often to update it (`interval`).

```
simbox = {  
    name      = "$JOBNAME$_replica$REPLICA$_simbox.dat"  
    first     = 0.0  
    interval  = 1.2  
}
```

B.4.21 energy_group

Desmond can record a break-down of the energy components during the simulation. The `energy_group` parameter controls this functionality and specifies the name, time to start recording (`first`) and frequency to record (`interval`) this information, e.g.,

```
energy_group = {  
    name      = "$JOBNAME$_replica$REPLICA$_enegrp.dat"  
    first     = 0.0  
    interval  = 1.2  
    self_energy = false  
    corr_energy = true  
}
```

Setting `corr_energy` to `true` allows the correction energy to be printed to the output file. Setting `self_energy` to `true` includes the self-energy term of the Ewald summation into the correction energy.

B.4.22 backend

The descriptions above handle most (if not all) of the parameters that you would typically use with Desmond. However, if additional parameters available in the back-end config file are needed then the `backend` parameter can be used to specify them in a front-end config file.

For example:

```
backend = {  
    force.nonbonded.r_lazy = 12.0  
}
```

adds `force.nonbonded.r_lazy = 12.0` to the back-end config file.

Settings within the backend maps have the highest precedence and thus are honored unconditionally. Incorrect settings within this map are not detected by the driver.

B.5 Parameters for Minimization

All common parameters can be set in a config file used for a minimization job without causing errors, but understandably, not every common parameter makes sense for minimization, e.g., annealing, temperature, timestep. Such parameters, if set in the config file, are ignored in minimization calculations.

B.5.1 max_steps

The `max_steps` parameter specifies the maximum number of steps (or iterations) that the Desmond minimizer runs, e.g.

```
max_steps = 2000
```

B.5.2 convergence

The `convergence` parameter sets the convergence criterion in kcal/(mol angstrom), e.g.,

```
convergence = 1.0
```

B.5.3 steepest_descent_steps

It is often good to use a few steepest descent minimization steps prior to using more elaborate minimization approaches. The keyword `steepest_descent_steps` sets the number of steepest descent steps to use, e.g.,

```
steepest_descent_steps = 10
```

After these steps, the LBFSG method is used to further minimize the system.

B.5.4 num_vector

This parameter sets the number of vectors to use for LBFSG method, e.g.

```
num_vector = 3
```

B.6 Parameters for Replica Exchange Simulations

All common parameters can be set in a config file used for a replica exchange simulation job without causing errors.

B.6.1 replica

The `replica` parameter sets the simulation configuration for each replica. There are two forms for setting this parameter. The first is for replica exchange calculations in which the temperature is set in each replica for all atoms in the system. Here, the `replica` parameter is a list, and each element of the list is a map value, which in turn specifies the simulation configuration for the corresponding replica. This example

```
replica = [ { temperature = 300.0 }
            { temperature = 302.0 }
            { temperature = 305.0 }
            { temperature = 308.0 }
            { temperature = 312.0 }
          ]
```

specifies that 5 replicas that will run, one at each of 300.0, 302.0, 305.0, 308.0, and 312.0 K. Within each of these map values, additional common parameters may be set. Note that there is no default value for replica parameters, since the settings often strongly depend on the model system.

Sometimes, different CMS files are used for different replicas. The CMS files can be specified in this form:

```
replica = [ { model_file = file1.cms  temperature = 300.0 }
            { model_file = file2.cms  temperature = 302.0 }
            { model_file = file3.cms  temperature = 305.0 }
            { model_file = file4.cms  temperature = 308.0 }
            { model_file = file5.cms  temperature = 312.0 }
          ]
```

The second form of the `replica` parameter is used for the REST (replica exchange with solute tempering) method [46], in which the temperature is set for only a subset of the atoms (the solute) in each replica. Here, `replica` is specified as a map, for example

```
replica = {
  generator = solute_tempering
  atom = "asl:atom.num 1-128"
  temperature = [300.0 400.0 500.0 600.0]
}
```

The `generator` keyword specifies that solute tempering is used to generate the replicas. The `atom` keyword specifies the atoms involved, and the `temperature` list provides the temperatures to be used.

The precedence of replica settings over other settings is:

- backend for replica
- backend
- replica
- common settings

B.7 Parameters for Metadynamics Simulations

All common parameters can be set in a config file used for a replica exchange simulation job without causing errors.

B.7.1 meta

The `meta` block is used to set up a metadynamics simulation. The parameters that apply to all collective variables (CVs) are the height of the repulsive Gaussian potential, in kcal/mol, the time in ps at which they are first added, and the interval in ps at which they are subsequently added. The collective variables are defined by the `cv` keyword.

```
meta = {
  cv = cv-map
  cv_name = $JOBNAME.cvseq
  first = 0.0
  height = 0.03
  interval = 1.2
  name = kernels.kerseq
}
```

Two output files are defined: `cv_name` sets the name of the metadynamics log file (default `$JOBNAME.cvseq`), and `name` sets the name of the metadynamics output file, which contains the Gaussian widths and height at each step, and is used to calculate the free energy. You can specify the time of the last addition with the `last` keyword; the default is the simulation time.

B.7.2 cv

The `cv` parameter defines the collective variables in a metadynamics simulation. Each variable is defined in a map that includes the type (`dist`, `angle`, `dihedral`, `rmsd`), the atoms that are in the collective variable, and the RMS width of the repulsive Gaussian potential. For distance variables, a wall can be placed at a specified distance, which prevents the system from moving

too far in the direction defined by the collective variable. For the analysis, the range of coordinate values can be defined. This is useful if the interesting phenomena occur at the end of the range, e.g. at 180° for a dihedral angle. An example with two CVs is as follows:

```
cv = [ { type = dist
        atom = [1 3]
        width = 0.4
        wall = 10.0
        { type = angle
          atom = [1 3 5 6]
          width = 0.4
          range = [0 360] } } ]
```

To define a collective variable in terms of the center of mass of a set of atoms, you can provide a list of atoms inside the `atom` list, e.g.

```
atom = [[2379 2380 2382 2384 2385] [631 642 651]]
type = "dist"
```

In this example, the first “atom” in the distance variable is the center of mass of six atoms, and the second “atom” is the center of mass of three atoms. Distance variables are in angstroms, angles are in degrees.

If you define the collective variable in the config file, you can only use a list of atoms to define the center of mass. If you want to use ASL to define the center of mass, you can do so in a meta block in the `.msj` file.

B.8 Parameters for `vruntime`

All common parameters that can be set in a config file can be used for a `vruntime` job without causing errors.

B.8.1 `vruntime_frameset`

The `vruntime_frameset` parameter is used to set the path of the MD simulation trajectory that should be analyzed. e.g.,

```
vruntime_frameset = /home/joeuser/Desmond_files/my_study_trj
```

There is no default value for this parameter.

Analyzing a Simulation from the Command Line

This appendix documents the command line usage of two related Python scripts and the syntax of two file formats used for analysis of simulation quality. The two Python scripts are `simulation_block_data.py` and `simulation_block_test.py`, and are located in the directory `$SCHRODINGER/mmshare-vversion/lib/Linux-x86/lib/python2.6/site-packages/schrodinger/application/desmond/`. These scripts can be run with the `$SCHRODINGER/run` command.

C.1 `simulation_block_data.py`

This command determines simulation properties from the input `.log` file and block averages from the input `.ene` file, and writes the results to the output `.sba` file. The syntax is as follows:

```
simulation_block_data.py [-n block-length] [-s simboxfile] -e enefile
                        -l logfile -c sbafile
```

The options are described in [Table C.1](#).

Table C.1. Options for the `simulation_block_data.py` command.

Option	Description
<code>-e enefile</code>	Input <code>.ene</code> file name from a simulation. No default.
<code>-l logfile</code>	Input <code>.log</code> file name from a simulation. No default
<code>-c sbafile</code>	Output <code>.sba</code> file name. No default.
<code>-n block-length</code>	Input block length in ps for calculation of block averages. Default: 10 ps.
<code>-s simboxfile</code>	Simbox <code>.dat</code> file name. This is the file that Desmond writes out to track how the shape of the simulation box evolves during the simulation.

C.2 simulation_block_test.py

This command evaluates the results in an input .sba file using tests specified in an .sbt file. The output file is a plain text file that prints the job details block from the .sba file and then provides information on whether block averages are within the tests specified in the .sbt file. The syntax is as follows:

```
simulation_block_test.py [-d] -i sbafile -t sbtfile -o outfile
```

The output file has the following information for each test in the .sbt file.

- Test (Pass or Fail)
- Testing criteria from input .sbt file
- Corresponding block values from input .sba file

An example output file has the following information besides the job details block from the .sba file.

```
Test for E: Fail
Testing criteria: SD = 5.000, Slope = 4.300 kcal/mol/ps , Average = -435320.200
+/- 4000.000 kcal/mol
Block data: SD = 44.941, Slope = -0.034 kcal/mol/ps, Average = -128666.563 kcal/
mol
```

C.3 Simulation Block Analysis (.sba) File Syntax

The .sba file contains properties obtained from the .log file and block averages from the .ene file. It has a header block followed by the job details block and then by data blocks.

The header block contains the information on the energy and log files and has the following format:

```
Version: version
Energy_File: enefile
Log_File: logfile
```

The job details block contains the information from the log file about the simulation, such as the status of the simulation, number of atoms, ensemble, and so on. An example block is shown below:

```
Block: Job_Details
Status = Normal
Temperature = 300.0
Job_name = rin
Degrees_of_freedom = 103139
Molecules = 3
```

```
Duration = 1.2
Atoms = 50274
Ensemble = MTK_NPT
End_Block
```

Subsequent data blocks printed in the .sba file are block averages enclosed between Block and End_Block lines. A sample data block is shown below:

```
Block: E
Time(ps) E(kcal/mol)
5.0 5.0
10.0 4.9
End_Block
```

The first row is a heading that indicates what quantities are listed below, and it is followed by rows of values. This block indicates that block average for E for data points up to the first 5 ps was 5.0 kcal/mol, and for the next 5 ps (5ps to 10ps) it was 4.9 kcal/mol.

C.4 Simulation Block Test (.sbt) File Syntax

Simulation block test (.sbt) files are used to test the data in .sba files and determine the stability of the simulation. These files contain the test parameters for various properties. A sample test set is as follows:

```
E {
sd = 5.0
slope = 4.3
average = -435320.2
average_tol = 4000.0
}
```

This block indicates that the block values for the property called E in an input .sba file should have following properties:

- Standard deviation < 5.0
- Slope (i.e. drift with respect to time) < 4.3
- Average should be within -435320.2 ± 4000

References

1. Bowers, K.J.; Chow, E.; Xu, H.; Dror, R. O.; Eastwood, M. P.; Gregerson, B. A.; Klepeis, J. L.; Kolossvary, I.; Moraes, M. A.; Sacerdoti, F. D.; Salmon, J. K.; Shan, Y.; Shaw, D. E. Scalable Algorithms for Molecular Dynamics Simulations on Commodity Clusters, Proceedings of the ACM/IEEE Conference on Supercomputing(SC06), Tampa, Florida, November 11-17, **2006**, http://sc06.supercomputing.org/schedule/event_detail.php?evid=9088.
2. Shaw, D.E. A fast, scalable method for the parallel evaluation of distance-limited pair wise particle interactions. *J. Comput. Chem.* **2005**, *26*, 1318.
3. Bowers, K.J.; Dror, R.O.; Shaw, D.E. The midpoint method for parallelization of particle simulations. *J. Chem. Phys.* **2006**, *124*, 184109.
4. Bowers, K.J.; Dror, R.O.; Shaw, D.E. Zonal methods for the parallel execution of range-limited N-body simulations. *J. Comput. Phys.* **2007**, *221*, 303.
5. Lippert, R.A.; Bowers, K.J.; Dror, R. O.; Eastwood, M.P.; Gregersen, B. A.; Klepeis, J. L.; Kolossvary, I.; Shaw, D. E. A common, avoidable source of error in molecular dynamics integrators. *J. Chem. Phys.* **2007**, *126*, 046101.
6. Arkin, I.T.; et al. Mechanism of Na⁺/H⁺ Antiporting. *Science*, **2007**, *317*, 799.
7. Humphrey, W.; Dalke, A.; Schulten, K. VMD - Visual Molecular Dynamics, *J. Molec. Graphics*, **1996**, *14*, 33.
8. Cornell, W.D.; Cieplak P.; Bayly, C. I.; Gould, I. R.; Merz, K. M.; Ferguson, D. M.; Spellmeyer, D. C.; Fox, T.; Caldwell, J. W.; Kollmann, P. A. *J. Am. Chem. Soc.* **1995**, *117*, 5179. Parameters converted from those at <http://amber.scripps.edu/amber9.ffparms.tar.gz>
9. Kollman, P. A. *Acc. Chem. Res.* **1996**, *29*, 461. Parameters converted from those at <http://amber.scripps.edu/amber9.ffparms.tar.gz>
10. Wang, J.; Cieplak, P.; Kollman, P. *J. Comput. Chem.* **2000**, *21*, 1049. Parameters converted from those at <http://amber.scripps.edu/amber9.ffparms.tar.gz>
11. Hornak et al. *Proteins: Structure, Function & Genetics*, **2006**, *3*, 712.
12. Shaw, D. E.; Maragakis, P.; Lindorff-Larsen, K.; Piana, S.; Dror, R. O.; Eastwood, M. P.; Bank, J. A.; Jumper, J. M.; Salmon, J. K.; Shan Y.; Wriggers, W. *Science* **2010**, *330*, 341.

13. Duan, Y.; Wu, C. Chowdhury, S; Lee, M. C.; Xiong, G.; Zhang, W.; Yang, R.; Cieplak, P.; Luo, R.; Lee, T.; Caldwell, J.; Wang, J.; Kollman, P. *J. Comput. Chem.* **2003**, *24*, 1999. Parameters converted from those at <http://amber.scripps.edu/amber9.ffparms.tar.gz>. Bugfix from <http://amber.scripps.edu/bugfixes/9.0/bugix.5> applied to correct torsional assignments.
14. Parameters generated from http://mackerell.umaryland.edu/CHARMM_ff_params_files/toppar/toppar_c35b2_c36a2.tar.gz.
15. Beglov, D.; Roux, B. *J. Chem. Phys.* **1994**, *100*, 9050 (ions).
16. MacKerell, A. D., Jr., et al. *J. Phys. Chem. B.* **1998**, *102*, 3586 (proteins).
17. MacKerell, Jr. A. D.; Feig M.; Brooks, A. D., III *J. Comput. Chem.* **2004**, *25*, 1400 (protein CMAP). Missing CMAP term applied to protonated HIS.
18. Foloppe N.; MacKerell, A. D., Jr. *J. Comp. Chem.* **2000**, *21*, 86 (nucleic acids).
19. MacKerell, A. D., Jr. Banavali, N. K. *J. Comp. Chem.* **2000**, *21*, 105 (nucleic acids).
20. Feller, S. E.; MacKerell, A. D., Jr. *J. Phys. Chem. B.* **2000**, *104*, 7510 (lipids).
21. Feller, S. E.; Gawrisch, K.; MacKerell, A. D., Jr. *J. Am. Chem. Soc.* **2002**, *124*, 318 (lipids).
22. Klauda, J. B.; Brooks, B. R.; MacKerell, A. D., Jr. *J. Phys. Chem. B*, **2005**, *109*, 5300 (alkanes/lipids).
23. Klauda, J. B.; Venable, R. M.; Freites, J. A.; O'Connor, J. W.; Tobias, D. J.; Mondragon-Ramirez, C.; Vorobyov, I.; MacKerell, A. D., Jr.; Pastor, R. W. **2010**, *114*, 7830.
24. Jorgensen, W. L.; Maxwell, D. S.; Tirado-Rives, J., *J. Am Chem. Soc.* **1996**, *118*, 11225.
25. Damm, W.; Frontera, A.; Tirado-Rives, J.; Jorgensen, W. L. *J. Comput. Chem.* **1997**, *18*, 1955.
26. Jorgensen, W. L., et al. *Theochem.* **1998**, *424*, 145.
27. McDonald, N. A., Jorgensen, W. L. *J. Phys. Chem. B.* **1998**, *102*, 8049.
28. Rizzo, R. C.; Jorgensen, W. L. *J. Am. Chem. Soc.* **1999**, *121*, 4827.
29. Watkins, E. K.; Jorgensen, W. L., *J. Phys. Chem. A.* **2001**, *205*, 4118.
30. Kaminski, G. A.; Friesner, R. A.; Tirado-Rives, J.; Jorgensen, W. L. *J. Phys. Chem. B* **2001**, *105*, 6474.
31. OPLSAA/L reparameterization, version 1 torsions used for SER, version 1 for ASP, version 3 (combined) for LEU, VAL, Jacobson, M.P., et al. *J. Phys. Chem. B.* **2002**, *106*,

11673. The charges for HISE used in this force field have not been published to our knowledge.
32. Jensen, K. P.; Jorgensen, W. L. *J. Chem. Theory Comput.* **2006**, 2, 1499.
 33. Jacobson, M.P.; Kaminski, G. A.; Friesner, R. A.; Rapp, C. S. *J. Phys. Chem. B.* **2002**, 106, 11673.
 34. Berendsen, H. J. C. et al. in *Intermolecular Forces*, edited by B. Pullman (Reidel, Dordrecht, 1981), p. 331.
 35. Berendsen, H. J. C.; Grigera, J. R.; Straatsma, T. P. *J. Phys. Chem.* **1987**, 91, 6269.
 36. Jorgensen, W. L.; Chandrasekhar, J.; Madura, J. D.; Impey, R. W.; Klein, M. L. *J. Chem. Phys.* **1983**, 79, 926. Parameters as tabulated in Mahoney, M. W.; Jorgensen, W. L. *J. Chem. Phys.* **2000**, 112, 8910.
 37. Neria, E.; Fischer, S.; Karplus, M. *J. Chem. Phys.* **1996**, 105, 1902.
 38. Jorgensen, W. L.; Madura, J. D. *Mol. Phys.* **1985**, 56, 1381. Parameters as tabulated in Mahoney, M. W.; Jorgensen, W. L. *J. Chem. Phys.* **2000**, 112, 8910.
 39. Horn, H. W.; Swope, W. C.; Pitara, J. W.; Madura, J. D.; Dick, T. J.; Hura, G. L. *J. Chem. Phys.* **2004**, 120, 9665.
 40. Mahoney, M. W.; Jorgensen, W. L. *J. Chem. Phys.* **2000**, 112, 8910.
 41. Earl D. J.; Deem, M. W.; *Phys. Chem. Chem. Phys.*, **2005**, 7, 3910.
 42. Guo Z; Mohanty U.; Noehre J.; Sawyer T. K.; Sherman W.; Krilov G. Probing the α -helical structural stability of stapled p53 peptides: molecular dynamics simulations and analysis. *Chem. Biol. Drug Des.* **2010**, 75, 348.
 43. Patriksson, A; van der Spoel, D., A temperature predictor for parallel tempering simulations. *Phys. Chem. Chem. Phys.* **2008**, 10, 2073.
 44. Gervasio, F. L.; Laio, A.; Parrinello, M. Flexible docking in solution using metadynamics. *J. Am. Chem. Soc.* **2005**, 127, 2600.
 45. Shivakumar, D.; Williams, J.; Wu, Y.; Damm, W.; Shelley, J.; Sherman, W. Prediction of Absolute Solvation Free Energies using Molecular Dynamics Free Energy Perturbation and the OPLS Force Field. *J. Chem. Theory Comput.* **2010**, 6, 1509.
 46. Liu, P.; Kim, B.; Friesner, R. A.; Berne, B. J., Replica exchange with solute tempering: a method for sampling biological systems in explicit water. *Proc. Natl. Acad. Sci. U.S.A.* **2005**, 102, 13749.

Getting Help

Information about Schrödinger software is available in two main places:

- The `docs` folder (directory) of your software installation, which contains HTML and PDF documentation. Index pages are available in this folder.
- The Schrödinger web site, <http://www.schrodinger.com/>, particularly the Support Center, <http://www.schrodinger.com/supportcenter>, and the Knowledge Base, <http://www.schrodinger.com/kb>.

Finding Information in Maestro

Maestro provides access to nearly all the information available on Schrödinger software.

To get information:

- Pause the pointer over a GUI feature (button, menu item, menu, ...). In the main window, information is displayed in the Auto-Help text box, which is located at the foot of the main window, or in a tooltip. In other panels, information is displayed in a tooltip.

If the tooltip does not appear within a second, check that **Show tooltips** is selected under **General → Appearance** in the Preferences panel, which you can open with **CTRL+, (⌘,)**. Not all features have tooltips.

- Click the **Help** button in a panel or press **F1** for information about a panel or the tab that is displayed in a panel. The help topic is displayed in your browser.
- Choose **Help → Online Help** or press **CTRL+H (⌘H)** to open the default help topic in your browser.
- When help is displayed in your browser, use the navigation links or search the help in the side bar.
- Choose **Help → Manuals Index**, to open a PDF file that has links to all the PDF documents. Click a link to open the document.
- Choose **Help → Search Manuals** to search the manuals. The search tab in Adobe Reader opens, and you can search across all the PDF documents. You must have Adobe Reader installed to use this feature.

For information on:

- Problems and solutions: choose Help → Knowledge Base or Help → Known Issues → *product*.
- Software updates: choose Maestro → Check for Updates.
- New software features: choose Help → New Features.
- Scripts available for download: choose Scripts → Update.
- Python scripting: choose Help → Python Module Overview.
- Utility programs: choose Help → About Utilities.
- Keyboard shortcuts: choose Help → Keyboard Shortcuts.
- Installation and licensing: see the *Installation Guide*.
- Running and managing jobs: see the *Job Control Guide*.
- Using Maestro: see the *Maestro User Manual*.
- Maestro commands: see the *Maestro Command Reference Manual*.

Contacting Technical Support

If you have questions that are not answered from any of the above sources, contact Schrödinger using the information below.

E-mail: help@schrodinger.com

USPS: Schrödinger, 101 SW Main Street, Suite 1300, Portland, OR 97204

Phone: (503) 299-1150

Fax: (503) 299-4532

WWW: <http://www.schrodinger.com>

FTP: <ftp://ftp.schrodinger.com>

Generally, e-mail correspondence is best because you can send machine output, if necessary. When sending e-mail messages, please include the following information:

- All relevant user input and machine output
- Desmond purchaser (company, research institution, or individual)
- Primary Desmond user
- Installation, licensing, and machine information as described below.

Gathering Information for Technical Support

This section describes how to gather the required machine, licensing, and installation information, and any other job-related or failure-related information, to send to technical support.

For general enquiries or problems:

1. Open the Diagnostics panel.
 - **Maestro:** Help → Diagnostics
 - **Windows:** Start → All Programs → Schrodinger-2012 → Diagnostics
 - **Mac:** Applications → Schrodinger2012 → Diagnostics
 - **Command line:** \$SCHRODINGER/diagnostics
2. When the diagnostics have run, click Technical Support.

A dialog box opens, with instructions. You can highlight and copy the name of the file.
3. Attach the file specified in the dialog box to your e-mail message.

If your job failed:

1. Open the Monitor panel in Maestro.

Use Applications → Monitor Jobs or Tasks → Monitor Jobs.
2. Select the failed job in the table, and click Postmortem.

The Postmortem panel opens.
3. If your data is not sensitive and you can send it, select Include structures and deselect Automatically obfuscate path names.
4. Click Create.

An archive file is created in your working directory, and an information dialog box with the name of the file opens. You can highlight and copy the name of the file.
5. Attach the file specified in the dialog box to your e-mail message.
6. Copy and paste any log messages from the window used to start Maestro (or the job) into the email message, or attach them as a file.
 - **Windows:** Right-click in the window and choose Select All, then press ENTER to copy the text.
 - **Mac:** Start the Console application (Applications → Utilities), filter on the application that you used to start the job (Maestro, BioLuminate, Elements), copy the text.

If Maestro failed:

1. Open the Diagnostics panel.

- **Windows:** Start → All Programs → Schrodinger-2012 → Diagnostics
- **Mac:** Applications → Schrodinger2012 → Diagnostics
- **Linux/command line:** \$SCHRODINGER/diagnostics

2. When the diagnostics have run, click Technical Support.

A dialog box opens, with instructions. You can highlight and copy the name of the file.

3. Attach the file specified in the dialog box to your e-mail message.

4. Attach the file `maestro_error.txt` to your e-mail message.

This file should be in the following location:

- **Windows:** %LOCALAPPDATA%\Schrodinger\appcrash
(Choose Start → Run and paste this location into the Open text box.)
- **Mac:** Documents/Schrodinger
- **Linux:** Maestro's working directory specified in the dialog box (the location is given in the terminal window).

5. On Windows, also attach the file `maestro.EXE.dmp`, which is in the same location as `maestro_error.txt`.

A

absolute solvation free energy.....	48
Advanced Ion Placement dialog box	16
Amber	
converting input files to CMS.....	99
force fields available via viparr	87
analysis	
metadynamics.....	64, 123
simulation events	62, 123
simulation quality	61
atom groups	
defining in multsim file	113
specifying	38
thermostat	33

B

barostat	
coupling style	34, 133
methods available	34
pressure.....	34, 133
barrier, metadynamics distance variables .	29, 141
bond, attachment	42
bonded time step	32, 134
box shape	11, 78, 116
box size	11, 78
box volume, minimizing	11, 76, 116
buffer distance	
in plane of membrane	78
solute to box boundary	11, 116

C

charged molecules, correction to solvation free energy	52
CHARMM, force fields available via viparr.....	87
checkpoint file.....	37, 106
config file keyword.....	136
importing	20
probing.....	71, 108
clashes, ligand-protein, relieving	43
collective variables.....	28
bin size for analysis	123
config file block.....	141
maximum number.....	29
range for analysis.....	123
command syntax	
amber_prm2cms	99

desmond.....	68
desmond_restraints.....	102
manipulate_trj.....	98
mold_gpcr_membrane.....	100
multsim.....	70
rebuild_cms	103
solvate_pocket.....	94
system_builder.....	75
trajectory_extract_frame.....	101
viparr.....	85
communication plugin	69
composite model system (CMS) files.....	80, 137
specifying for replicas	140
composite system builder (CSB) file.....	75
example	76
multsim keyword.....	116
config file.....	127
back-end, adding keywords for	138
macros	128
reading.....	19
syntax	127
units	128
configuration file– <i>see</i> config file	
conventions, document	ix
coordinate origin, resetting	116
corrections	
charged molecules, solvation free energy... ..	52
enthalpic, for restraints	124
long-range dispersion, for absolute free energy	124
coupling scheme, barostat.....	133
cutoff radius, nonbonded interactions.....	35, 131

D

desmond command	68
directory	
installation	6
Maestro working.....	6
dispersion correction, long-range	124
domain decomposition.....	39, 105
config file specification	135

E

energy components, config file keyword.....	138
energy recording	22
config file keyword.....	136

- ensemble
 - classes available..... 23
 - config file keyword..... 133
 - replica exchange 27
 - valid classes 133
- environment variable
 - SCHRODINGER 6
- Ewald tolerance..... 35
- F**
 - far time step..... 32, 134
 - FEP calculations 41
 - use of solvate_pocket..... 121
 - FEP protocol 50
 - adjusting 53
 - default 51
 - files
 - analysis report..... 123
 - checkpoint 106
 - deleting unwanted stage data..... 105, 125
 - extern stage extra 123
 - input, writing 20, 42
 - multisim (msj) 105
 - output structure 111
 - relaxation protocol..... 24
 - structure, for FEP jobs 118
 - force fields 14, 79
 - available with viparr 87
 - merging 86
 - modifying 85
 - specifying with viparr..... 86
 - system builder assignment..... 116
 - free energy
 - average as function of time..... 125
 - plot, as function of collective variables 64
 - relative binding 41, 46, 51
 - solvation 41, 48, 52
- G**
 - Gaussian potential, metadynamics..... 28, 141
 - glue..... 37
 - config file keyword..... 135
- I**
 - images
 - plotter to use 123, 124
 - saving trajectory frame..... 61
 - input files, writing..... 20, 42
 - integration algorithm, setting parameters for.... 32
 - interactions
 - Coulomb method, config file keyword..... 132
 - short-range and long-range..... 35
 - truncation at cutoff 35, 131
 - interval
 - averaging, for simulation quality analysis.. 61
 - metadynamics potential addition..... 141
 - velocity randomization 137
 - interval, recording
 - checkpoint file 37, 136
 - energy 22, 136
 - energy group..... 138
 - simulation box 138
 - trajectory..... 22, 135
 - ions
 - adding 14, 79, 116
 - excluding from region 15, 79
 - placing near residues 15, 79
 - types available 15
- J**
 - job name..... 39
 - macro for, in config file..... 129
 - multisim job..... 71
 - jobs, restarting multisim 106
- L**
 - lambda schedule..... 54
 - config file keywords 130
 - default for absolute free energies 131
 - default for relative free energies 130
 - lambda windows 54
 - index 130
 - structure files 118
 - ligand environment, FEP simulations 49
 - ligand fragments 43
 - clashes with protein 43
 - editing 44
 - Ligand Functional Group Mutation by FEP
 - panel 43
 - ligand mutation 42
 - long-range dispersion correction for absolute free energy 124

M

master job	39
host	71
membrane	
adding	11, 78
adjusting orientation	13
available models	11
buffer distance	78
embedding GPCR in	100
OPM	12
placing	12
relaxation protocol	24, 30
surface area	30
surface tension	30
metadynamics	28
analysis	64
bin size for analysis	123
config file block	141
distance barrier	29, 141
log file name	141
mutlsim block	119
Metadynamics Analysis panel	65
Metadynamics panel	29
minimization	21
algorithm	21
convergence threshold	22, 139
maximum iterations	21, 139
number of LBFGS vectors	139
setting parameters for	34
steepest descent steps	139
Minimization panel	21
model system	
adding ions to	79
adding salt to	80
importing	21
neutralizing	80
preparing	75
relaxing	23
selecting	20
solvating	80
Molecular Dynamics panel	22
msg file, example	72
multisim	70, 105
file syntax	109
job file example	72
restarting jobs	106
template commands	105

mutation FEP

ligand	42
protein residue	46
ring atom	44
mutation FEP, use of Hamiltonian replica	
exchange	51, 55

N

near time step	32, 134
neutralizing the model system	80
node locking	106

O

output, config file keyword	137
-----------------------------------	-----

P

partial charges	13
particle mesh Ewald (PME) method	132
periodic boundary	
setting up	11, 78
wrapping molecules at	135
pressure	
config file keywords	133
simulation, setting	23
probability profiles	123
processors	
allocation	39
choosing number of	39, 70
config file keyword	135
product installation	152
properties, for simulation event analysis	62
protein side chains, mutating	46

R

random seed	
Bennett method, FEP analysis	124
velocity	137
recording	
checkpoint file	37, 136
energy	22, 136
energy group	138
simulation box	138
start time	37
trajectory	22, 135
update frequency	37

- relative binding free energy 41
- relaxation protocol
 - default 23
 - membrane 30
 - modifying 24
- replica exchange 27
 - config file temperature keyword 140
 - default ensemble 27
 - graph of exchanges 63
 - mutation FEP use 51, 55
 - solute tempering (REST) 140
 - temperature 27, 69
 - temperature profile 27
- Replica Exchange Dynamics Review panel 63
- Replica Exchange Graph panel 64
- report file 123
- restarting jobs
 - from a completed stage 107
 - with different config file 107
 - with new stages 107
- restarting subjobs
 - maximum attempts 71
- restraints
 - enthalpic correction for 124
 - on atom positions 36
- S**
- salt, adding 16, 80
- Schrödinger contact information 152
- Set Up Membrane panel 12
- side chain
 - modifying 48
 - reorienting 48
- simulated annealing
 - activating, config file keyword 133
 - number of stages 25
 - temperature schedule, config file
 - keyword 132
- Simulated Annealing panel 25
- simulation event analysis 62
 - specification block, MSJ file 123
- Simulation Event Analysis panel 63
- simulation parameters
 - extra for desmond command 69
- Simulation Quality Analysis panel 62
- simulation time 22
 - config file keyword 134
- solutes
 - keeping together 37, 135
 - reading for model system 76
- solvation free energy 41
 - correcting for charged molecules 52
- solvent
 - models 10
 - multisim keyword 116, 117
 - specifying 9, 77
- stages, multisim job 110
 - common keywords 111
 - deleting unwanted data files 125
 - Desmond 111
 - inserting on restart 107
 - modifying on restart 107
 - restarting from completed 107
 - skipping 111
 - supported 110
- Start dialog box 40
- start time
 - checkpoint file recording 37, 136
 - energy group recording 138
 - energy recording 136
 - metadynamics potential addition 141
 - simulation 134
 - simulation box recording 138
 - trajectory 135
 - velocity randomization 137
- structures
 - checking validity of 4
 - preparation 4, 9
 - reading solute for system building 77
 - specifying for viparr 86
- subjobs
 - changing stage parameters 107
 - host to run on 71
 - job name 111
 - location 39, 71, 106
 - maximum number 71, 107
 - maximum restarts 71
 - subdirectory names 111
 - synchronizing 111
- substitution group, ligand mutation 42
- surface area, membrane 30
- surface tension
 - config file keyword 133
 - simulation, setting 23
- system builder 75

System Builder panel	
Ions tab	14
Solvation tab	10

T

tapering schemes	35, 131
temperature	
config file keyword	132
desmond command option for replica exchange	69
randomized velocity target	137
replica exchange	27, 140
simulated annealing, config file keyword	132
simulation, setting	23
solvate_pocket	120
thermostat	132
temperature profile, replica exchange	27
temperature program	24
thermodynamic properties	
plot over time	62
thermostat	
atom groups	38
temperature	132
thermostat groups	33
thermostat method	33
time	
free energy as function of	125
simulation start	134
total simulation	134
time series, variables for analysis	123
time steps	32
config file keyword	134
topology, processor	39, 70, 105, 135
Total Free Energy by FEP panel	49
trajectory	
combined	98
config file keyword	135
directory	24, 37
frame image	61
frame positioning	59
frames per file	135
path to	142
recording interval	22
replica exchange	28
saving as movie	61
saving structures from	61
smoothing during play	59

superimposing on Workspace structure	60
viewing during simulation	61
viewing in Maestro	57
viewing with VMD	81
visible atoms	60
Trajectory panel	58
transmembrane atoms, defining	12

U

units, in config file	128
update frequency	37

V

velocities	
randomizing	38, 137
recording in trajectory	37

W

water	
force-field models	87
moving out of solute entry	116
penetration between protein and membrane	30

120 West 45th Street
17th Floor
New York, NY 10036

155 Gibbs St
Suite 430
Rockville, MD 20850-0353

Quatro House
Frimley Road
Camberley GU16 7ER
United Kingdom

101 SW Main Street
Suite 1300
Portland, OR 97204

Dynamostraße 13
D-68165 Mannheim
Germany

8F Pacific Century Place
1-11-1 Marunouchi
Chiyoda-ku, Tokyo 100-6208
Japan

245 First Street
Riverview II, 18th Floor
Cambridge, MA 02142

Zeppelinstraße 73
D-81669 München
Germany

No. 102, 4th Block
3rd Main Road, 3rd Stage
Sharada Colony
Basaveshwaranagar
Bangalore 560079, India

8910 University Center Lane
Suite 270
San Diego, CA 92122

Potsdamer Platz 11
D-10785 Berlin
Germany

SCHRÖDINGER®