Python Pipeline Manual

Schrödinger Software Release 2015-2

Schrödinger Press

May 2015

Contents

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	`$SCHRODINGER/maestro`	File names, directory names, commands, environment variables, command input and output
Italic	*filename*	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].

Overview

1.1 The Python Pipeline

The Python pipeline is a framework for running jobs that consist of multiple stages that can involve multiple products. This framework consists of a set of Python modules that perform various tasks and make use of the underlying libraries that are used by the products themselves as necessary. The framework for running the jobs includes capabilities for distributing the jobs and for restarting the jobs.

The stages and the input for the stages, including communication between stages, are specified in the input file. This file is described in detail in Chapter 2. Information on running the jobs is given in Chapter 3.

The pipeline is used for running several workflows: the Virtual Screening Workflow (VSW), Quantum-Polarized Ligand Docking (QPLD), Prime Covalent Docking, and Phase database creation and management. These workflows can be run from Maestro. You can customize these workflows by writing out the input file (click Write in the panel), editing the file, then running the job. You can also create your own workflows by assembling the stages that you want to use in the input file, and running the job.

1.2 Running Schrödinger Software

Schrödinger applications can be run 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. The Maestro interface is a customization of the Maestro interface. You can also use the standard Maestro as your working interface.

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 the Maestro interface. This directory then becomes the 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-2015-2 → 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.

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 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-2015-2. 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 SchrodingerSuite2015-2 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. To set other variables, on OS X 10.7 use the command

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

and on OS X 10.8, 10.9, and 10.10 use the command

`launchctl setenv` *variable* `"`*value*`"`

The Input File

In this chapter, the input file is described in detail. Normally, you would use the Write button in one of the Maestro panels to write the input file, and then modify it for your purposes. However, you can construct an input file without using a file written from Maestro.

2.1 Input File Syntax

The input file consists of a number of blocks. Each block has the following syntax

```
[ blocktype:blockname ]
    keyword   value
    ...
    keyword   value
```

A block is terminated by the start of another block or the end of file. The keyword is separated from the value by a space. Otherwise, spaces and blank lines are not significant, but we recommend that you add them for readability. Keywords and their values are case-sensitive.

In addition to the blocks, comments can be added by preceding them with a # sign. Any text following a # sign is treated as a comment, unless the # sign is inside single or double quotes. Thus, the # in `"ligand#1"` is not treated as the beginning of a comment because it is inside a set of quotes.

The input file must have one or more `SET` blocks, which are used to specify the input files and other data, followed by one or more `STAGE` blocks, which specify the program (module) and its data, and be terminated by a `USEROUTS` block, which specifies the files to return to the launch directory.

The `SET` block defines a variable that can be assigned a value. This variable can then be used as a value for a keyword later in the file. The variable name is the blockname value for the `SET` block. These variables are typically used to define input files. Setting variables allows you to easily reuse the input file and just change the variable values to run a calculation on another system.

The class of variable is defined by the keyword, `VARCLASS`, which can be followed by one of several keywords that defines the value of the variable. These keywords may be valid only for certain classes. The keywords and their dependencies are described in Table 2.1.

Table 2.1. Keywords for the SET block.

Keyword	Description
FILES	Comma-separated list of file names. Can be used with Grid, Structures, Text classes.
PATH	Path to a directory. Used only with PhaseDB class.
VARCLASS	Variable class. Allowed values: Grid, PhaseDB, Structures, Text.

An example from the VSW workflow is shown below:

```
[ SET:ORIGINAL_LIGANDS ]
    VARCLASS Structures
    FILES /zone1/me/glide/tutorial/structures/50ligs_epik.mae.gz
```

This file can be referred to in any of the stages by the name ORIGINAL_LIGANDS.

The STAGE block defines a stage in the pipelined workflow, in which a particular task is performed. The available stages and the keywords that are specific to each stage are defined in the next section. The stage name (value of *blockname*) is a label for the stage. At the top of each stage, there must be a STAGECLASS keyword, which defines the module that is being executed and the particular stage from that module. It must be followed by an INPUTS keyword and an OUTPUTS keyword, which define the input and output files for the stage. These files are defined by an internal name, not by the actual file name.

For example, the first stage in a VSW workflow could be a LigPrep stage, for which the top of the block could look like the following:

```
[ STAGE:LIGPREP ]
    STAGECLASS              ligprep.LigPrepStage
    INPUTS                  ORIGINAL_LIGANDS
    OUTPUTS                 LIGPREP_OUT
```

Here, the module name is ligprep, and the stage name is LigPrepStage. The input structure file is labeled ORIGINAL_LIGANDS, and in this case is the file defined in the SET block above. The output structure file is labeled LIGPREP_OUT. The actual file name is set by the program using a default naming convention. In the next stage, LIGPREP_OUT can appear as the value for the INPUTS keyword, as follows:

```
[ STAGE:POSTLIGPREP ]
    STAGECLASS              ligprep.PostLigPrepStage
    INPUTS                  LIGPREP_OUT
    OUTPUTS                 POSTLIGPREP_OUT
```

In this manner, files can be passed from one stage to another. If there are multiple files, they are given in a comma-separated list, as in the following example:

```
[ STAGE:DOCK_HTVS_1 ]
    STAGECLASS              glide.DockingStage
    INPUTS                  LIPFILTER_OUT, GRID_1
    OUTPUTS                 HTVS_OUT_1
```

The USEROUTS block has two keywords, USEROUTS, which specifies the output files to return, and STRUCTOUT, which specifies the structure files to return. These are comma-separated lists of the values given for OUTPUTS in one or more of the stages.

2.2 Available Modules and Stages

In this section each module is described, and the keywords for each stage available in the module are tabulated. In addition to the tabulated keywords, each stage must have an INPUTS and an OUTPUTS keyword.

2.2.1 Module combine

This module combine multiple structure file sets into a single file.

2.2.1.1 Stage CombineStage

This stage combines multiple structure sets into one set, with the option of labeling each ligand by the input set from which it came. The stage takes up to ten input structure files sets and generates one output structure file set. The keywords are described in Table 2.2.

Table 2.2. Keywords for the CombineStage stage of the combine module.

Keyword	Description
LABELFIELD	Name of optional property added to each output structure that holds the label for the input set from which it came. The property must be in the format of a Maestro internal name for a string property, s_*family_name*.
LABELS	Comma-separated list of labels for the input sets. Default: None

2.2.1.2 Stage GenerateCovalentComplexesStage

This stage creates covalent complexes by forming a covalent bond between each of a set of ligands and a receptor. with a set of ligands via covalent bond to form complexes. The atom on the receptor to which each ligand is bonded must be specified, along with the receptor atom bonded to this atom that is part of the leaving group. The ligand leaving group is specified by a

list of SMARTS patterns and the atom number in the SMARTS pattern of the atom that is attached to the receptor. This stage is used by Prime Covalent Docking (see *Covalent Docking*). The keywords are described in Table 2.3.

Table 2.3. Keywords for the GenerateCovalentComplexesStage stage of the combine module.

Keyword	Description
RECEP_STAYING_ATOM	Atom number in the receptor to bond the ligand to. Required.
RECEP_LEAVING_ATOM	Atom number in receptor of atom to replace when making a bond.
SMARTS	SMARTS patterns that match the leaving group in the ligand, as a comma-separated list.
POSITIONS	Atom index for each SMARTS pattern of the ligand atom to attach to the receptor, as a comma-separated list. There must be one index for each SMARTS pattern listed for the SMARTS keyword.
MARK_LIGAND	Set the chain and residue number of the ligand in the complex to this value. Default: X:1.

2.2.2 Module convert

This module converts structure files from one format to another. It has a single stage.

2.2.2.1 Stage ConvertStage

This stage converts structure files from one format to another. The formats that are supported are Maestro, PDB, SMILES, either as a SMILES string with an optional title (.smi) or in a CSV file with properties (.csv), and SD (.sd). The file type is determined by the extension. Files whose extensions are not recognized are treated as SD files. Not all conversions are supported. The supported conversions are summarized in Table 2.4.

Table 2.4. Supported format conversions.

From \ To	Maestro	PDB	SD	SMILES	CSV
Maestro		Yes	Yes	Yes	Yes
PDB	Yes		No	No	No
SD	Yes	No		Yes	Yes
SMILES	Yes	No	No		Yes
CSV	Yes	No	No	No	

Multiple input files can be specified in a single run of this stage. All input files are converted to a single format. The list of structure files to be converted must be defined in a SET block first, and then the set name passed as the argument to the INPUTS keyword. The output file names are constructed from the value *setname* of the OUTPUTS keyword as *setname*-*NNN*.*ext*, with the extension determined by the output file format. The keywords are described in Table 2.5.

Table 2.5. Keywords for the ConvertStage stage of the convert module.

Keyword	Description
2D	Generate 2D SD structures even if the input is 3D.
OUTFORMAT	Output file format. Allowed values: smiles, smilescsv, sd, maestro. Default: maestro.

2.2.3 Module glide

This module runs Glide grid generation and ligand docking calculations. It is used by VSW. Many of the keywords are the same as in the Glide input file.

2.2.3.1 Stage GridgenStage

This stage generates the receptor grid for Glide docking calculations. The keywords are described in Table 2.6. Correspondences between the Glide input file keywords (given in Table 7.3 of the *Glide User Manual*) and the GridgenStage keywords are listed in the table. The default values are the Glide defaults unless otherwise specified.

Table 2.6. Keywords for the GridgenStage stage of the glide module .

Keyword	Description
RECEP_VSCALE	General van der Waals radius scaling factor. Same as Glide input file.
RECEP_CCUT	General van der Waals radius scaling partial charge cutoff. Same as Glide input file.
GRID_CENTER	*x*, *y*, and *z* coordinate for grid center.
XCENT	*x* coordinate for grid center. Obsolete - use GRID_CENTER instead.
YCENT	*y* coordinate for grid center. Obsolete - use GRID_CENTER instead.
ZCENT	*z* coordinate for grid center. Obsolete - use GRID_CENTER instead.
INNERBOX	Dimensions of inner box as comma-separated list. Same as Glide input file.
OUTERBOX	Dimensions of outer box as comma-separated list. Same as Glide input file.
INNERBOXX	*x* dimension for inner box.

Table 2.6. Keywords for the GridgenStage stage of the glide module (Continued).

Keyword	Description
INNERBOXY	*y* dimension for inner box.
INNERBOXZ	*z* dimension for inner box.
OUTERBOXX	*x* dimension for outer box. Same as ACTXRANGE in Glide input file.
OUTERBOXY	*y* dimension for outer box. Same as ACTYRANGE in Glide input file.
OUTERBOXZ	*z* dimension for outer box. Same as ACTZRANGE in Glide input file.
DELETEWATERS	Delete all waters. Default: False.
WRITEZIP	Write out compressed grids. Default: False.
HBOND_CONSTRAINTS	List of H-bond constraints. Each constraint has the format *label atomnum*. Default: No H-bond constraints.
METAL_CONSTRAINTS	List of metal constraints. Each constraint has the format *label atomnum*. Default: No metal constraints.
NOE_CONSTRAINTS	List of NOE constraints. Each constraint has the format: *label x y z min-radius max-radius*. Default: No NOE constraints.
POSIT_CONSTRAINTS	List of positional constraints. Each constraint has the format: *label x y z radius*. Default: No positional constraints.

2.2.3.2 Stage DockingStage

This stage runs a docking calculation with predefined grids. The INPUTS keyword must specify the ligand file first, followed by the grid file. The stage keywords are described in Table 2.7. Most of the keywords are the same as in the Glide input file, but not all Glide input file keywords are supported.

Table 2.7. Keywords for the DockingStage stage of the glide module.

Keyword	Description
Settings keywords	
PRECISION {XP|SP|HTVS}	Docking precision mode.
DOCKING_METHOD {confgen|rigid|mininplace|inplace}	Docking method: confgen—Dock flexibly. rigid—Dock rigidly. mininplace—Refine (do not dock) inplace—Score in place (do not dock)
SAMPLE_RINGS *boolean*	Sample rings if set to True and DOCKING_METHOD is set to confgen.

Table 2.7. Keywords for the DockingStage stage of the glide module. (Continued)

Keyword	Description
AMIDE_MODE {penal|free|fixed|trans}	Amide bond sampling mode. Only set when DOCKING_METHOD is set to confgen. penal—penalize nonplanar conformation free—vary conformation fixed—retain original conformation trans—allow trans conformation only
AMIDE_TRANSTOL *angle*	Maximum angle deviation in degrees from 180° for an amide to be considered trans. Default: 20.
EPIK_PENALTIES *boolean*	Apply penalties for ionization or tautomeric states calculated by Epik. Default: False.
WRITE_XP_DESC *boolean*	Write XP descriptor information if set to True. Default: False.
MAXKEEP *nposes*	Number of poses per ligand to keep in initial phase of docking. Default: 5000.
SCORING_CUTOFF *cutoff*	Scoring window for keeping initial poses. Default: 100.0.
MAXREF *nposes*	Number of poses to keep per ligand for energy minimization. Default: 400.
DIELECTRIC *constant*	Distance-dependent dielectric constant. Default: 2.0.
FORCEFIELD {OPLS2001|OPLS2005}	Specify the force field to use. Glide is parametrized against the OPLS_2001 force field. Default: OPLS2001.
FORCEPLANAR *boolean*	Scale force-field parameters for planar (sp^2) systems such as rings to strongly penalize nonplanar conformations. Default: False.
MAX_ITERATIONS *niter*	Maximum number of conjugate gradient steps. Default:100.
Ligands keywords	
LIGFORMAT	Ligand file format. Allowed values: maestro, sd, mol2, pdb. Default: derive from extension.
LIGAND_START *firstlig*	First ligand in ligand file to dock. Default: 1.
LIGAND_END *lastlig*	Last ligand in ligand file to dock. A value of 0 means the last ligand in the file. Default: 0.
LIG_MAECHARGES *boolean*	Take charges from input structures if set to True. Default: False.
MAXATOMS *maxatoms*	Maximum number of atoms per ligand. Default (and maximum): 300.
MAXROTBONDS *maxrotbonds*	Maximum number of rotatable bonds per ligand. Default (and maximum): 50
LIG_VSCALE *factor*	Scaling factor for van der Waals radii scaling. Default: 0.8.

Table 2.7. Keywords for the DockingStage stage of the glide module. (Continued)

Keyword	Description
`LIG_CCUT` *cutoff*	Partial charge cutoff for van der Waals radii scaling. Default: 0.15.
`RINGCONFCUT` *cutoff*	Energy window for retention of ring conformers. Default: 2.5 kcal mol^{-1}.
Core keywords	
`USE_REF_LIGAND` *boolean*	Use core pattern for comparison or restraint if set to `True`. Default: `False`.
`REF_LIGAND_FILE` *filename*	File name for reference ligand. File format can be Maestro, SD, MOL2 or PDB.
`CORE_DEFINITION {all|smarts|allheavy}`	Specify how the core is defined. `all`—use all atoms in the reference ligand `allheavy`—use all nonhydrogen atoms in the reference ligand `smarts`—use the SMARTS pattern defined by the `CORE_ATOM` keyword.
`CORE_ATOMS` *list*	List of core atoms to use for the RMSD calculation. Only set when `CORE_DEFINITION` is set to `smarts`.
`CORE_SMARTS` *pattern*	Smarts pattern for core. Only set when `CORE_DEFINITION` is set to `smarts`. Set in the Core tab.
`CORE_RESTRAIN` *boolean*	Restrict docking to reference position if set to `True`. Default: `False`.
`CORE_RESTRAIN_V` *value*	Strength of core restraining potential. Default: `5.0`.
`CORE_POS_MAX_RMSD` *value*	Tolerance on the RMSD to use when restricting docking to the reference position.
`CORE_FILTER` *boolean*	Skip ligands that do not match the core pattern if set to `True`. Default: `False`.
Constraints keywords	
`[ CONSTRAINT_GROUP:`*n*` ]`	Define constraint group *n*. The square brackets are mandatory, because this is a section heading that must be followed by other keywords. For pipeline jobs, a second pair of square brackets must be added.
`USE_CONS` *list*	List of labels of the constraints that are to be used in docking. Used in a constraint group definition; can also be used independently for defining constraints. If the constraint requires a feature definition, you can append to the label a colon and the index of the feature definition defined by the `FEATURE` keyword, e.g. `position1:1`. Default: no constraints.

Table 2.7. Keywords for the DockingStage stage of the glide module. (Continued)

Keyword	Description
NREQUIRED_CONS *n*	Number of constraints that are required to match, or ALL. Used in a constraint group definition; can also be used independently for defining constraints. Default: require all constraints in the group.
[FEATURE:*n*]	Define custom feature *n*. The square brackets are mandatory, because this is a section heading that must be followed by other keywords. For pipeline jobs, a second pair of square brackets must be added.
PATTERN*n* *smarts atoms* [include|exclude]	Pattern to be used for a custom feature, given as a SMARTS pattern (*smarts*) followed by a list of atom indices in the SMARTS pattern that must match (*atoms*). By default, atoms matching the pattern are included in the feature. You can specify whether to include atoms in the feature or exclude them from the feature by adding include or exclude.
Output keywords	
COMPRESS_POSES {TRUE|FALSE}	Compress the output pose files. Default: TRUE.
CV_CUTOFF *cutoff*	Reject poses with Coulomb-van der Waals energy greater than *cutoff* kcal/mol. Default: 0.0.
HBOND_CUTOFF *cutoff*	Reject poses with H-bond score greater than *cutoff*. Default: 0.0.
METAL_CUTOFF *cutoff*	Reject poses with metal score greater than *cutoff*. Default: 10.0.
GSCORE_CUTOFF *cutoff*	Reject poses with GlideScore greater than *cutoff*.
POSE_RMSD	RMS deviation used in clustering to discard poses, in angstroms. Default: 0.5.
POSE_OUTTYPE {poseviewer|ligandlib}	Type of output file to produce. Default: poseviewer. poseviewer—Write pose viewer file (_pv.mae); includes receptor as first structure. ligandlib—Write ligand pose file (_lib.mae); does not contain receptor.
POSE_DISPLACEMENT	Maximum atomic displacement used in clustering to discard poses, in angstroms. Default: 1.3.
NREPORT *maxposes*	Maximum number of poses to write out from the docking run. Default: 10000.
POSES_PER_LIG *maxperlig*	Maximum number of poses to write per ligand. Default: 1.
NUM_TO_KEEP	Number of compounds to keep after the docking run. Default: 1.

Table 2.7. Keywords for the DockingStage stage of the glide module. (Continued)

Keyword	Description
PERCENT_TO_KEEP	Percentage of compounds to keep after the docking run. Default: 10.0
BEST_BY_TITLE	Keep only the best scoring pose per compound after the docking run. Default: False.
POSTDOCK *boolean*	Use post docking minimization if set to True. Default: True.
POSTDOCK_NPOSE *npose*	Number of poses to use in post-docking minimization. Maestro sets this number to 10 for XP. Default: 5.
POSTDOCKSTRAIN *boolean*	Apply strain correction terms if set to True. Default: False.
WRITE_RES_INTERACTION *boolean*	Generate per-residue interaction terms if set to True. Default: False.
RADIUS_RES_INTERACTION *r*	Radius for per-residue interaction terms, in angstroms. Only set if WRITE_RES_INTERACTION is set to True. Default: 12.0
WRITEREPT *boolean*	Write report text file (.rept for docking and .scor for score-in-place). Default: False.
VSW keywords	
RECOMBINE	Recombine ligand files. Default: True.
UNIQUEFIELD	Default: s_m_title.
COMPRESS_IN_FILES	Compress subjob input files. Default: True.
Deprecated keywords	
LVDW	Use LIG_VSCALE instead. .Default: 0.8.
LIGCCUT	Use LIG_CCUT instead. Default: 0.15.
LIGAND_CONFS	Use DOCKING_METHOD instead.

2.2.3.3 Stage GlideSortStage

This stage is used for sorting Glide pose files. The keywords are described in Table 2.8.

Table 2.8. Keywords for the GlideSortStage stage of the glide module.

Keyword	Description
BLOCK_SORT_BY_TITLE	Sort the poses by title. All poses with the same title are grouped and then sorted by the INTRATITLE_SORT_KEYS keys. The blocks themselves are sorted by applying the SORT_KEYS to the top pose in each block. Default: True.
INTRATITLE_SORT_KEYS	List of properties to use as sort keys within a title block. Multiple sort keys must be separated by white space. Default: r_i_glide_emodel.
SORT_KEYS	List of properties to use as sort keys for sorting the poses. If BLOCK_SORT_BY_TITLE is True, the blocks are sorted by the values for the top pose. Multiple sort keys must be separated by white space. Default: r_i_docking_score.
OUTPUT_POSES_PER_TITLE	The number of top poses to keep for each title block. Applies only if BLOCK_SORT_BY_TITLE is True. 0 means retain all poses. Default: 0.
INPUT_TYPE	Specify whether the receptor is (pv) or is not (lib) in the file. Default: determine the type from the file suffix, _pv.mae or _lib.mae.

2.2.3.4 Stage MergeStage

This stage merges Glide output files with the utility glide_ensemble_merge, so that the poses are in increasing value of GlideScore. The files must be already sorted by GlideScore. Up to 100 files can be merged. The output is a single pose file.

The keywords are described in Table 2.9.

2.2.4 Module gencodes

Generate and store codes for ligands/compounds and variants.

2.2.4.1 Stage GenCodesStage

The keywords are described in Table 2.10.

Table 2.9. Keywords for the MergeStage stage of the glide module.

Keyword	Description
NREPORT	Specify the number of poses to save. The default is 0, which retains all poses. Default: 0.
MAXPERLIG	Specify the maximum number of poses to keep for each ligand (determined by the structure title). Use 0 to retain all poses. Default: 1.
OFFSETS	List of GlideScore offsets to be applied to the input structures, one for each file. Default: 0.0 for all files.
MARKERS	List of string labels for each receptor, one for each file. Each ligand is marked by this value. Default: no label.

Table 2.10. Keywords for the GenCodesStage stage of the gencodes module

Keyword	Description
UNIQUEFIELD	Field unique to each compound. Default: NONE.
OUTVARIANTFIELD	Field to put variant codes into. Default: s_vsw_variant.
OUTFORMAT	Output file format. Default: mae
SKIP_BAD_LIGANDS	Default: True
SKIP_RECEPTOR	Do not include the receptor in the output file. Applies only to pose viewer files. Default: True.

2.2.4.2 Stage RestoreTitlesStage

This stage restores the original titles to the compounds. It is necessary because some stages modify the title for internal tracking purposes.

Table 2.11. Keywords for the RestoreTitlesStage stage of the gencodes module.

Keyword	Description
SOURCE_FIELD	Name of property in which the title is stored temporarily. Default: s_pipeline_title_backup.
DESTINATION_FIELD	Name of property to which the title is to be copied. Default: s_m_title.

2.2.5 Module filtering

The filtering module provides stages for filtering structures given a set of filtering criteria.

2.2.5.1 Stage ChargeFilterStage

The chargefilter module filters ligands based on total molecular charge. Ligands within the charge range are kept; other ligands are filtered out. The keywords for the ChargeFilterStage stage are described in Table 2.12.

Table 2.12. Keywords for the ChargeFilterStage stage of the chargefilter module.

Keyword	Description
MIN_CHARGE	Minimum charge on ligands that are kept. Required.
MAX_CHARGE	Minimum charge on ligands that are kept. Default: None.

2.2.5.2 Stage DrugLikeSplitStage

This stage splits the input ligand structures into 3 groups: those that pass the drug-like filter, which are written to *jobname*-dl.maegz; those that pass the coarse filter, which are written to *jobname*-co.maegz; and those that are left over, which are written to *jobname*-lo.maegz. If one variant (conformer, tautomer, stereoisomer) of a ligand passes all criteria of a particular filter, all variants are included in the output file for that filter.

The variants for a given compound root must be contiguous in the input file and have the same title.

This stage has no stage-dependent keywords.

2.2.5.3 Stage LigFilterStage

This stage runs a filtering job using LigFilter utility. The stage takes one set of input structure files and generates one set of corresponding output files. For more information on LigFilter, see Section 2.4 of the *General Utilities* manual.

The keywords are described in Table 2.13. One of these keywords must be used.

Table 2.13. Keywords for the LigFilterStage stage of the filtering module.

Keyword	Description
FILTER_FILE	File name for a Ligfilter criteria file.
CONDITIONS	Comma-separated list of LigFilter criteria. Ignored if FILTER_FILE is specified.

2.2.5.4 Stage GeneratePropsStage

This stage calculate properties and feature counts derived from the structures, and writes out the structures with the properties. The properties are calculated using the `-addprop` option to `ligfilter`—see Section 2.4 of the *General Utilities* manual.

This stage has no stage-dependent keywords.

2.2.5.5 Stage RemoveDuplicatesStage

This stage removes duplicate variants for each compound based on unique SMILES strings. The variants for a given compound root must be contiguous in the input file and have the same title.

This stage has no stage-dependent keywords.

2.2.5.6 Stage SubSetStage

Stage for making a subset of the input files based on a list of titles or value of another property. The stage takes one set of input structure files and generates one set of corresponding output files.

The keywords are described in Table 2.14.

Table 2.14. Keywords for the SubSetStage stage of the filtering module.

Keyword	Description
`PROPERTY`	Property to filter on. SMILES format files can only be filtered on the title. Default: `s_m_title`.
`VALUE_FILE`	File containing a list of values to keep (one per line) if a FILTER_FILE is specified.

2.2.6 Module ligprep

Run Ligprep and manage variants.

2.2.6.1 Stage LigPrepStage

This stages runs LigPrep on the input structures. The keywords are described in Table 2.15.

2.2.6.2 Stage PostLigPrepStage

This stage limits the number of stereoisomers, removes states penalized by Epik or ionizer, and generates a property that identifies the variants (tautomers, ionization states, stereoisomers, conformers) of each compound.

Table 2.15. Keywords for the LigPrepStage stage of the ligprep module.

Keyword	Description
RETITLE	Default: False.
UNIQUEFIELD	Default: NONE
STEREO_SOURCE	Source of stereo information. Allowed values: parities, geometry. Default: parities.
USE_EPIK	Use Epik for generating ionization and tautomeric states. Default: False.
METAL_BINDING	Generate metal-binding states with Epik. Default: False.
RETAIN	Retain input variant for each compound. Default: True.
PH	Target pH value. Default: 7.0.
PHT	Threshold for range of pH values. Default: 2.0.
IONIZE	Generate ionization states when using Ionizer (ionization states are always generated when using Epik). Default: True.
GENERATE_TAUTOMERS	Generate tautomers. Default: True.
MAX_TAUTOMERS	Maximum number of tautomers to generate when using Ionizer for ionization and tautomerizer for tautomers. Default: 8.
NEUTRALIZE	Neutralize before expanding states. Default: False.
MAX_STATES	Maximum number of states to generate with Epik, including both ionization and tautomeric states. Default: 16.
NUM_STEREOISOMERS	Maximum number of stereoisomers to generate per input structure. Default: 32.
NRINGCONFS	Maximum number of ring conformers to generate per ligand. Default: 1.
MIXLIGS	Default: False.
RECOMBINE	Recombine ligand files. Default: True.
COMBINEOUTS	Combine output files. Default: False.
SKIP_BAD_LIGANDS	Skip ligands that cannot be processes. Default: True.
SKIP_NOUNIQUE_LIGANDS	Skip ligands that have no unique field. Default: False.
TAUT_SPEC_FILE	Custom tautomer specification file to use.
OUTFORMAT	Output file format. Allowed values: mae, sd. Default: mae.

The keywords are described in Table 2.16.

Table 2.16. Keywords for the PostLigPrepStage stage of the ligprep module.

Keyword	Description
LIMIT_STEREOISOMERS	Limit the number of stereoisomers to the number specified by MAXSTEREO. Default: True.
MAXSTEREO	Maximum number of stereoisomers to keep, in order of energy. Default: 4.
REMOVE_PENALIZED_STATES	Remove states that are penalized by Epik or the Ionizer. Default: True.
UNIQUEFIELD	Name of property that is used to specify which structures are considered as belonging to the same compound. Default: s_m_title.
OUTVARIANTFIELD	Name of property to which the variant label is written. The label consists of the property given by UNIQUEFIELD and the index of the variant. Default: s_vsw_variant.
OUTFORMAT	Output file format. Allowed values: sd, mae. Default: mae.
PRESERVE_NJOBS	Write out as many output files as there are input files. Useful to create the same number of subjobs for QikProp as were used for LigPrep. Default: False.

2.2.7 Module macromodel

This module runs MacroModel conformational search and ring sampling jobs.

2.2.7.1 Stage ConfSearchStage

The keywords are described in Table 2.17.

Table 2.17. Keywords for the ConfSearchStage stage of the macromodel module.

Keyword	Description
FORCE_FIELD	Force field to use. Default: OPLS_2001
SOLVENT	Solvent model. Default: None
ELECTROSTATIC_TREATMENT	Electrostatic treatment. Default: Constant dielectric
DIELECTRIC_CONSTANT	Value of the dielectric constant. Default: 1.0
CHARGES_FROM	Source of charges. Default: use charges from the force field.
CUTOFF	Default: None
MINI_METHOD	Minimization method. Default: PRCG

Table 2.17. Keywords for the ConfSearchStage stage of the macromodel module. (Continued)

Keyword	Description
OUTCONFS_PER_SEARCH	Default: 1
CONFSEARCH_METHOD	Conformational search method. Default: MCMM
CONFSEARCH_STEPS	Total number of search steps. Default: 200
CONFSEARCH_STEPS_PER_ROTATABLE	Number of search steps per rotatable bond. Default: 50
MAXIMUM_ITERATION	Maximum iterations for minimization. Default: 500
CONVERGE_ON	Convergence criterion. Default: Gradient
CONVERGENCE_THRESHOLD	Minimization convergence threshold. Default: 0.05
ENERGY_WINDOW	Energy window for conformer elimination. Conformations that are higher in energy than the lowest-energy conformation by this amount are discarded. Default: 5.0
SERIAL_SPLIT_OUTPUT	Default: False
MULTI_LIGAND	Default: `True`. Set to `False` if the input file contains one set of conformers.

2.2.7.2 Stage SampleRingsStage

The keywords are described in Table 2.18.

Table 2.18. Keywords for the SampleRingsStage stage of the macromodel module.

Keyword	Description
FORCE_FIELD	Force field to use. Default: `OPLS_2005`.
SOLVENT	Solvent model. Allowed values: Default: Water.
ELECTROSTATIC_TREATMENT	Electrostatic treatment. Allowed values: Default: Constant dielectric
CHARGES_FROM	Source of charges. Allowed values: Default: Force field.
CUTOFF	Default: Extended
OUTCONFS_PER_SEARCH	Number of output conformers per input structure. Default: 1.
MAXIMUM_ITERATION	Maximum iterations for minimization. Default: 500
CONVERGE_ON	Convergence criterion. Default: Gradient.

Table 2.18. Keywords for the SampleRingsStage stage of the macromodel module. (Continued)

Keyword	Description
CONFSEARCH_STEPS_PER_ROTATABLE	Number of search steps per rotatable bond. Default: 10.
CONVERGENCE_THRESHOLD	Convergence threshold.Default: 0.05.
CONFSEARCH_STEPS	Total number of search steps. Default: 100.

2.2.8 Module prime

This module is used to run Prime MM-GBSA and structure refinement jobs. It is used for covalent docking jobs.

2.2.8.1 Stage MMGBSAStage

Stage for running Prime MM-GBSA on input structures. First structure in the input set is assumed to be the receptor. The stage takes one input structure file set and generates one set of corresponding output structure files (also in pose viewer format).

The keywords are described in Table 2.19.

Table 2.19. Keywords for the MMGBSAStage stage of the prime module.

Keyword	Description
USE_MAE_CHARGES	Use atomic partial charges from the input Maestro file rather than the force field. Default: False.
OUTPUT_LIG_STRAING	Write out an estimate of the ligand strain energy. Default: False.
USE_MEMBRANE	Use Prime implicit membrane model in protein and complex simulations. Default: False.

2.2.8.2 Stage PrimeStage

This stage runs a structure refinement job, using refinestruct. Many of the keywords are the same as for the prime input file, which is described in Section 10.2 of the *Prime User*

Manual. The keywords are listed in Table 2.20.

Table 2.20. Keywords for the PrimeStage stage of the prime module.

Keyword	Description
PRIME_TYPE	Same as refinestruct keyword. Allowed values: SIDE_PRED, LOOP_BLD, REAL_MIN, SITE_OPT, ENERGY, COVALENT_DOCKING.
THREADS	*
MAX_JOBS	*
NUM_OUTPUT_STRUCT	Same as refinestruct keyword.
ECUTOFF	Same as refinestruct keyword.
USE_RANDOM_SEED	Same as refinestruct keyword.
SEED	Same as refinestruct keyword.
USE_CRYSTAL_SYMMETRY	Same as refinestruct keyword.
INT_DIEL	Same as refinestruct keyword.
EXT_DIEL	Same as refinestruct keyword.
USE_MAE_CHARGES	Same as refinestruct keyword.
USE_MEMBRANE	Same as refinestruct keyword.
SELECT	Same as refinestruct keyword.
RESIDUE_FILE	*
LIGAND	Same as refinestruct keyword.
NPASSES	Same as refinestruct keyword.
LOOP_i_RES_j	Same as refinestruct keyword.
RES_SPHERE	Same as refinestruct keyword.
MAX_CA_MOVEMENT	Same as refinestruct keyword.
MIN_OVERLAP	Same as refinestruct keyword.
RESIDUE_i	Same as refinestruct keyword.
INCLUDE_RESIDUE	*

2.2.9 Module pull

This module extracts a subset of ligands or compounds from a file. It is used by VSW.

2.2.9.1 Stage PullStage

This stage identifies a subset of compounds from one set of ligand structures, and extracts all variants of those compounds from a second set of ligand structures, given the number or percent to keep from the first set. The first set must be ordered, that is, the ligands to keep must appear earliest in the set. Compounds in the second set that do not appear in the first set are ignored. This stage can operate on multiple input files.

Keeping the first *N* compounds may not choose compounds from Glide results in the proper manner. If the variants of a compound are chemically distinct, it is appropriate to choose the top compounds based on GlideScore. If the variants differ only in their conformations (such as when saving multiple poses per ligand), Emodel (not GlideScore) should be used to determine the representative variant of the compound for GlideScore comparison with other compounds.

The keywords are described in Table 2.21.

Table 2.21. Keywords for the PullStage stage of the pull module.

Keyword	Description
`UNIQUEFIELD`	The ligand property that identifies a compound. Multiple ligands with the same value of this keywords are considered variants of the compound. Default: `s_m_title`
`NUM_TO_KEEP`	Number of compounds from the first set to extract from the second set. The compounds that are kept are those that appear earliest in the first set. Default: `0`
`PERCENT_TO_KEEP`	Percentage of compounds from the first set to extract from the second set. The compounds that are kept are those that appear earliest in the first set. Default: `0.0`
`KEEP_CHARGES`	Boolean to save partial charges. Default: `False`

2.2.10 Module qikprop

This module runs QikProp jobs.

2.2.10.1 Stage QikPropStage

This stage runs QikProp jobs. QikProp properties are added to the output structures. The stage takes one set of input structure files and generates one set of corresponding output files. The number of subjobs is equal to the number of input files. There are no keywords for this stage.

2.2.11 Module qsite

This module runs QSite jobs. It is used by QPLD.

2.2.11.1 Stage QSiteStage

This stage runs a QSite job. The QM region must be defined in terms of ligands or ions: it cannot be defined by hydrogen caps or cuts in the receptor. The input structure file (`INPUTS`) must be a Maestro file containing one complex or a Glide pose-viewer file. For a pose-viewer file, the complexes are constructed as part of the stage, and can be distributed over multiple processors. Most of the stage keywords are used to set keywords in the QSite input file. These keywords are described in Section 5.3 of the *QSite User Manual* and Section 9.5 of the *Jaguar User Manual*. The keywords are listed in Table 2.22.

Table 2.22. Keywords for the QSiteStage stage of the qsite module.

Keyword	Description
`OUTPUT_LIGS_ONLY`	Write out only the ligands to the output file. This keyword is only valid if the input file is a pose-viewer file. Default: `False`.
`IGNORE_RECEP`	Ignore the receptor in the QM/MM calculation, and perform a QM calculation on the free ligand. This keyword is only valid if the input file is a pose-viewer file. Default: `False`.
`SOLVATION`	Use solvation. Sets the **isolv** keyword in the **gen** section of the input file to 2, and the **solvation_method** keyword in the **mmkey** section of the input file. Default: `False`.
`QM_MOLECULES`	List of molecule numbers to add to the QM region. Required if input is a complex; optional if input is a pose-viewer file. Default: None.
`QM_HCAP`	Not currently supported.
`QM_VSHIFT`	Sets the **vshift** keyword in the **gen** section of the input file.
`QM_MAXITG`	Sets the **maxitg** keyword in the **gen** section of the input file.
`QM_BASIS`	Sets the **basis** keyword in the **gen** section of the input file.
`QM_HCAPESCHG`	Not currently supported.
`QM_MAXIT`	Sets the **maxit** keyword in the **gen** section of the input file.
`QM_MULTIP`	Sets the **multip** keyword in the **gen** section of the input file.
`QM_MOLCHG`	Sets the **molchg** keyword in the **gen** section of the input file.
`QM_IGEOPT`	Sets the **igeopt** keyword in the **gen** section of the input file.
`QM_DFTNAME`	Sets the **dftname** keyword in the **gen** section of the input file.
`QM_MMQM`	Sets the **mmqm** keyword in the **gen** section of the input file.
`QM_IACC`	Sets the **iacc** keyword in the **gen** section of the input file.
`QM_IACSCF`	Sets the **iacscf** keyword in the **gen** section of the input file.

Table 2.22. Keywords for the QSiteStage stage of the qsite module. (Continued)

Keyword	Description
QM_ICFIT	Sets the **icfit** keyword in the **gen** section of the input file.
MM_USE_NB_CUTOFF	Sets the **use_nb_cutoff** keyword in the **mmkey** section of the input file.
MM_UPDATE_FREQ	Sets the **update_freq** keyword in the **mmkey** section of the input file.
MM_TN_FORCE_UPDATE	Sets the **tn_force_update** keyword in the **mmkey** section of the input file.
MM_MAXCYCLES	Sets the **maxcycles** keyword in the **mmkey** section of the input file.
MM_DELECTRIC	Sets the **dielectric** keyword in the **mmkey** section of the input file.
MM_NB_CUTOFF_DIST	Sets the **nb_cutoff_dist** keyword in the **mmkey** section of the input file.
MM_INIT_STEP_SIZE	Sets the **init_step_size** keyword in the **mmkey** section of the input file.
MM_MAX_STEP_SIZE	Sets the **max_step_size** keyword in the **mmkey** section of the input file.
MM_TN_FORCE_CUTOFF	Sets the **tn_force_cutoff** keyword in the **mmkey** section of the input file.
MM_BUF_FORCE_CONST	Sets the **buf_force_const** keyword in the **mmkey** section of the input file.
MM_DELTAE	Sets the **deltae** keyword in the **mmkey** section of the input file.
MM_GRADIENT	Sets the **gradient** keyword in the **mmkey** section of the input file.
MM_PARAMSTD	Obsolete.
MM_ESTATICS	Sets the **estatics** keyword in the **mmkey** section of the input file.
MM_OPT_METHOD	Sets the **opt_method** keyword in the **mmkey** section of the input file.
MM_CONVERGE_ON	Sets the **converge_on** keyword in the **mmkey** section of the input file.
MM_FORCEFIELD	Sets the **forcefield** keyword in the **mmkey** section of the input file.

2.2.12 Module rmsd

This module can be used to calculate RMSDs between structures in two sets.

2.2.12.1 Stage RmsdStage

This stage calculates the RMSD between structures in a reference set and structures in a second set, and writes out a file containing the RMSD as a property. The keywords are described in

Table 2.23.

Table 2.23. Keywords for the RmsdStage stage of the rmsd module.

Keyword	Description
UNIQUEFIELD_REF	Property in the reference set that is used to match structures for comparison. The RMSD is calculated for pairs in which the UNIQUEFIELD_REF value for a structure in the reference set matches the UNIQUEFIELD value for a structure in the second set. Default: the property set by the UNIQUEFIELD keyword.
UNIQUEFIELD	Property in the second set that is used to match structures for comparison. Default: s_m_title.
RMSDFIELD	Name of the RMSD property in the output file. Default: r_vsw_rmsd.
RENUMBER_STRUCTURES	Renumber the atoms before calculating the RMSD. This property should not be set if the atom numbering is the same for each pair of structures. Default: False.
IGNORE_STATES	Ignore hydrogens and treat all bonds as single bonds. Default: False.

Running the Pipeline

3.1 The Pipeline Job Command

The generic command for running the pipeline is

`$SCHRODINGER/run pipeline_startup.py [`*options*`] `*jobname*`.inp`

For specific workflows, you can use the top-level script that runs the workflow, for example:

`$SCHRODINGER/vsw [`*options*`] `*jobname*`.inp`

The progress of a pipeline job is as follows. First, the main driver starts on the specified host. This driver then starts the driver for each of the stages in succession. Each stage driver either performs a task directly, or runs a subjob to perform the specified tasks. When the stages have finished, the main driver performs any necessary final operations, and exits.

The drivers all run on the same host. By default this is the local host. You can run the drivers on a remote host either by setting `-REMOTEDRIVER`, in which case the drivers run on the first host in the `-HOST` list, or by explicitly specifying a host with `-DRIVERHOST`. The drivers can be idle for most of the duration of the job, because most of the work is done by the subjobs. It is therefore usually not advisable to submit the driver to a host such as a cluster, where resources are in high demand. However, if you do submit the driver to a queue, you should select a queue on which it can run until the entire job finishes.

Several options are provided for specifying the hosts on which to run subjobs. If the number of licenses you have is limited, you can specify different numbers of hosts for each program that is run with the `-host_`*program* options.

3.2 Restarting Pipeline Jobs

If a pipeline job fails for some reason, it can be restarted at a point that is related to where the failure occurred, provided that the relevant files are available. To ensure that you can restart a job, you must use the `-LOCAL` option, so that the subjob files are written to a disk system that is accessible to the drivers.

When you restart a job, the job must be run from the same directory as the original job. You can use the same command as you used to start the original job, and you must use the same input file to restart the job. You do not need to specify any options, but if you do, values that

differ from the original job are used instead of the original value. For example, to restart a VSW job, use the command:

```
$SCHRODINGER/vsw jobname.inp
```

Before the job is started, you are asked if you want to restart the job, or overwrite it with a new job run from the beginning. You can bypass this prompt with command options:

- If you want to force a job to be restarted, use the `-RESTART` option.
- If you want to overwrite a job, use the `-OVERWRITE` option.

When a subjob that failed for a particular stage is rerun, it starts from the beginning, rather than the point of failure. For this reason, it is useful to create a larger number of smaller subjobs. However, the number should not be too large, or time will be wasted on keeping track of the subjobs.

Getting Help

Schrödinger software is distributed with documentation in PDF format. If the documentation is not installed in `$SCHRODINGER/docs` on a computer that you have access to, you should install it or ask your system administrator to install it.

For help installing and setting up licenses for Schrödinger software and installing documentation, see the *Installation Guide*. For information on running jobs, see the *Job Control Guide*. Check the following sources for other information:

- The knowledge base, at http://www.schrodinger.com/kb
- Known Issues pages, at https://www.schrodinger.com/knownissues.

The manuals are installed in the `docs` directory of the software installation.

If you have questions that are not answered from any of the above sources, contact Schrödinger using the information below.

Web:	http://www.schrodinger.com/supportcenter
E-mail:	help@schrodinger.com
Mail:	Schrödinger, 101 SW Main Street, Suite 1300, Portland, OR 97204
Phone:	+1 503 299-1150 (USA, 9am – 5pm Pacific Time)
	+49 621 438-55173 (Europe, 9am – 5pm Central European Time)
Fax:	+1 503 299-4532 (USA, Portland office)
FTP:	ftp://ftp.schrodinger.com

Generally, using the web form is best because you can add machine output and upload files, if necessary. You will need to include the following information:

- All relevant user input and machine output
- Purchaser (company, research institution, or individual)
- Primary user
- Computer platform type
- Operating system with version number
- Version numbers of products installed
- mmshare version number

120 West 45th Street
17th Floor
New York, NY 10036

101 SW Main Street
Suite 1300
Portland, OR 97204

245 First Street
Riverview II, 18th Floor
Cambridge, MA 02142

8910 University Center Lane
Suite 270
San Diego, CA 92122

155 Gibbs St
Suite 430
Rockville, MD 20850-0353

Dynamostraße 13
D-68165 Mannheim
Germany

Zeppelinstraße 73
D-81669 München
Germany

Potsdamer Platz 11
D-10785 Berlin
Germany

Quatro House
Frimley Road
Camberley GU16 7ER
United Kingdom

8F Pacific Century Place
1-11-1 Marunouchi
Chiyoda-ku, Tokyo 100-6208
Japan

No. 102, 4th Block
3rd Main Road, 3rd Stage
Sharada Colony
Basaveshwaranagar
Bangalore 560079, India

SCHRÖDINGER®