General Utilities

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].

General Utilities

This document describes the general utilities that are available in the `utilities` directory of the installation. Utilities that are part of a specific product are described with that product. Utilities are intended to run in a Unix shell, and the syntax and descriptions are given on this basis. The syntax descriptions give only the utility name. On Linux or Mac, if `$SCHRODINGER/utilities` is not in your path, you should prefix the command with `$SCHRODINGER/utilities/`. On Windows, you can open a Schrodinger Command Prompt window from the Start menu to run utilities. In this window, the `utilities` directory is already in your path.

For information on command options, run the utility command with the `-h` option.

A summary of all utilities in the `utilities` directory is given in the *Schrödinger Utilities* quick reference sheet.

1 Structure Format Conversion

This section describes the general utilities available for conversion between various file formats for molecular structures. These utilities are used by Maestro for structure conversion when importing or exporting structures. Canvas also has a structure conversion utility, `canvasConvert`, for conversion between Maestro, SD, and SMILES/CSV format.

Some of these utilities allow a range of structures to be specified. A range specification is a comma-separated list of indices or ranges, with no spaces. A range is defined by the endpoints of the index range, separated by a colon. Examples of valid range specifications are:

1,4	structures 1 and 4
1:10,14	structures 1 through 10 and 14
2:	structures 2 through the end of file
:5,13:18	structures 1 through 5 and 13 through 18

1.1 Conversion Between Various Formats: structconvert

This utility converts between Maestro, MDL SD, PDB, Sybyl Mol2, SMILES, and MacroModel format files. The entire contents of the file are converted, with the exception that only the first structure is written when writing to PDB format. The syntax is as follows:

`structconvert` [*options*] [`-i`[*format*]] *inputfile* [`-o`[*format*]] *outputfile*

where *format* can be one of the following:

`mae`	Maestro format
`mae1`	Maestro version 1 format (pre-2013). Output only.
`mae2`	Maestro version 2 format (2013 on). Output only.
`sd`	V2000 SDfile format
`mm`	MacroModel (`.dat`) format (input only)
`pdb`	PDB format
`mol2`	Sybyl (`.mol2`) format
`smi`	SMILES format
`csv`	CSV file with SMILES for structure

If the `-i` and `-o` format options are omitted, the file extensions are used to determine the format, which can include conversion between compressed and uncompressed versions of the same file format.

The format conversions have the following limitations:

- PDB and SMILES format cannot be interconverted.
- Only the first structure is written when writing to PDB format.
- Structures cannot be written to the obsolete MacroModel `.dat` format.
- SMILES cannot be directly converted to Maestro format.

For conversion to and from PDB format, `structconvert` supports all the options that are supported by `pdbconvert`.

1.2 Merging Files with Format Conversion: structcat

This utility merges several files of the same type and writes them out to a single file, with format conversion if the file type is different. If the output file type is PDB, multi-structure input files are split into separate PDB files, rather than merged. The syntax of the command is:

`structcat -i`*format inputfile* `[-i`*format inputfile*`2 ...] [-o`*format*`]` *outputfile*

where *format* can be one of the following:

`mae`	Maestro format
`sd`	V2000 SDfile format
`pdb`	PDB format
`mol2`	Sybyl (`.mol2`) format
`smi`	SMILES format

Conversions are performed with the utilities described here, except that SMILES conversion is done with libraries from Canvas. If the output format is omitted, the format is determined from the file extension. Compressed files can be used, and different input file formats can be used.

1.3 Conversion To and From PDB Format: pdbconvert

This section describes the command-line utility version of the program that converts files between PDB, MacroModel, and Maestro formats. For information on PDB conversion within the Maestro GUI, see Section 3.1.7 on page 49. The syntax of the command is:

`pdbconvert` [*options*] `-i`*fmt inputfile* `-o`*fmt outputfile*

where *fmt* is one of `pdb`, `mae` (Maestro) or `mm` (MacroModel). Either the input or output file must be a PDB file—or both, for PDB-to-PDB conversion. Compressed PDB (`.pdb.gz`, `.pdbgz`, `.ent.gz`, `.entgz`) and Maestro (`.mae.gz`, `.maegz`) files are supported.

In the converted file, the title property is set to the PDB ID (`s_m_title` in Maestro files), and the PDB title is stored in a new property, PDB_TITLE (`s_pdb_PDB_TITLE` in Maestro files). HET, HETNAME, and FORMUL records are written to PDB files, and the het residue name and formula are written to Maestro files. Experimental temperature and pH data is read from and written to EXPDTA and associated REMARK records in PDB files, and stored in Maestro files as properties `s_pdb_PDB_EXPDTA_`*n*, `r_pdb_PDB_EXPDTA_`*n*`_TEMPERATURE_`*m*, and `r_pdb_PDB_EXPDTA_`*n*`_PH_`*m*. Here *n* indexes experimental data sets and *m* indexes values used in a given data set. These suffixes are absent if there is only one experiment or value.

1.3.1 Conversion Behavior for Importing PDB

Three sets of criteria are used for the placement of bonds in the conversion from PDB: a set of standard residue templates, the CONECT records of the PDB file, and geometry. Nonstandard PDB conventions are also interpreted to assist in bond placement. Bonds to metals are converted to zero-order bonds, and the formal charges are adjusted to represent the bonds as ionic bonds.

Where multiple atomic coordinates exist for a single PDB entry, the atoms with the highest occupancy ratio are read by default. This choice can be altered with the `-occ` option. The `-first_occ` and `-all_occ` options are equivalent to `-occ all` and `-occ first`, but the `-occ` usage is preferred.

A few amino acids require choices to be made for placement of double bonds and formal charges. Schrödinger's conventions for these choices are given in Table 1.

Table 1. Placement of double bonds and formal charges for imported amino acids.

Amino Acid Pair	Double Bond	Formal Charge
ARN/ARG	between CZ and NH1	for ARG, +1 formal charge on NH1
ASP/ASH	between CG and OD1	for ASP, -1 formal charge on OD2
GLU/GLH	between CD and OE1	for GLU, -1 formal charge on OE2

Note: PDB files that have varying number of atoms in different MODELs cannot be converted and result in a fatal error.

1.3.2 Conversion Error Codes for pdbconvert

When running `pdbconvert` from UNIX, you will receive numerical messages (2, 1, 0) indicating the status of an attempted PDB conversion. These messages are equivalent to Maestro's dialog box warnings. The message numbers are defined as follows:

2 (ERROR): A fatal error occurred, and no Maestro file was generated. Check disk permissions and disk space. This failure can also occur if the PDB file has varying number of atoms in different MODELs.

1 (WARNING): A Maestro file was created, but an error at or above the base error level was returned. At the default base error level, this value is returned when red or blue atoms are present, and for orange atoms if the duplicate CONECT record convention is not followed. See Table 3.2 of the *Maestro User Manual* for a description of the atom colors.

0 (OK): A Maestro file was generated without errors at or above the base error level.

1.3.3 Using Templates for Nonstandard Residues

The `pdbconvert` program obtains connectivity and bond order information for standard residues from a standard residue template file. Version 2 (unremediated) and Version 3 (remediated) PDB formats are both supported, in files at the following locations:

```
$SCHRODINGER/mmshare-vversion/data/mmpdb/mmpdb_v2.ini.
$SCHRODINGER/mmshare-vversion/data/mmpdb/mmpdb_v3.ini.
```

For nonstandard residues, the program predicts connectivity, but not bond order. However, if the connectivity and bond order information for a nonstandard residue is known, it can be used to read in the PDB file. To provide information, create a new file that contains a template with the required information. The following example for the alanine residue from the standard template illustrates the format:

```
TEMPLATE{
"ALA "
" N  " " CA " 1
" N  " " H  " 1
" CA " " CB " 1
" CA " " C  " 1
" C  " " O  " 2
" HA " " CA " 1
"1HB " " CB " 1
"2HB " " CB " 1
"3HB " " CB " 1
}
```

A connectivity template must begin with `TEMPLATE{` and end with a closing brace `}`. The first line in the template specifies the residue name. Each subsequent line in the template should specify two 4-character atom names, followed by a bond order. Templates for ligand molecules can also be created. To specify the template file to be used by `pdbconvert`, use the `-data` option.

1.3.4 Correcting Specific Errors

Specific errors in standard PDB structures, such as missing bonds, extraneous bonds or incorrect bond orders, can be corrected by using the errata initialization file, which is read by `pdbconvert`. This file is named `mmpdb_errata.ini`, and is stored in `$SCHRODINGER/mmshare-v`*version*`/data/mmpdb`. The file is in JSON format—for information and documentation, go to http://www.json.org. Many text editors provide a JSON model to help keep the hierarchy correct. You can supply your own errata file to supplement and override errata in the standard file, by naming it `pdb_errata.ini` and placing it in the current directory (searched first) or in `$HOME/.schrodinger/mmshare` (searched next).

The file is structured with blocks (objects) for each type of erratum. The available erratum types are `DELETE_BONDS` and `ADD_BONDS`. The example below shows a `DELETE_BONDS` block.

```
{
  "DELETE_BONDS" : {
             "1ABE" : [
                     { "SERIAL_OF_ATOM_1" : "2318" , "SERIAL_OF_ATOM_2" : "2337" },
                     { "SERIAL_OF_ATOM_1" : "2319" , "SERIAL_OF_ATOM_2" : "2334" },
                     { "SERIAL_OF_ATOM_1" : "2320" , "SERIAL_OF_ATOM_2" : "2335" },
                     { "SERIAL_OF_ATOM_1" : "2321" , "SERIAL_OF_ATOM_2" : "2336" },
                     { "SERIAL_OF_ATOM_1" : "2322" , "SERIAL_OF_ATOM_2" : "2337" },
                     { "SERIAL_OF_ATOM_1" : "2323" , "SERIAL_OF_ATOM_2" : "2328" },
                     { "SERIAL_OF_ATOM_1" : "2324" , "SERIAL_OF_ATOM_2" : "2329" },
                     { "SERIAL_OF_ATOM_1" : "2325" , "SERIAL_OF_ATOM_2" : "2330" },
```

```
                    { "SERIAL_OF_ATOM_1" : "2326" , "SERIAL_OF_ATOM_2" : "2331" },
                    { "SERIAL_OF_ATOM_1" : "2327" , "SERIAL_OF_ATOM_2" : "2328" },
                     { "SERIAL_OF_ATOM_1" : "2327" , "SERIAL_OF_ATOM_2" : "2332" }
                      ],
                }
}
```

Spaces are not significant, but line breaks are. To add an erratum for a new protein, you must add an object for that protein to the appropriate block. For `DELETE_BONDS`, the format is as follows:

```
"PDB-ID" : [
    { "SERIAL_OF_ATOM_1" : "sn1" , "SERIAL_OF_ATOM_2" : "sn2" }
           ],
```

while for `ADD_BONDS` (used for both adding and changing the bond order), the format is as follows:

```
"PDB-ID" : [
    { "SERIAL_OF_ATOM_1" : "sn1" , "SERIAL_OF_ATOM_2" : "sn2", "BOND_ORDER" : order }
           ],
```

Here, *PDB-ID* is the 4-character PDB ID of the protein, *sn1* and *sn2* are the atom numbers of the two atoms that are bonded or to be bonded, and *order* is the bond order, which is an integer (1, 2, or 3). The atom number is the number from the PDB file, and must be enclosed in quotes. You can include more than one bond in the block for a protein by duplicating the specification in braces, one per line—see the example above. To add an erratum for an existing protein, simply add the specification in braces for the relevant bond. The comma following the closing brace or square bracket is required unless it is the last object in the block.

Errata can only be applied to remediated PDB files (v3.0 and later). If you want to correct an unremediated file (v2.3 or earlier) you should convert it first.

1.4 Conversion To and From SD Format: sdconvert

Convert between MDL SD, Maestro, and MacroModel format files. The syntax is:

`sdconvert` [*options*] `-i`*fmt inputfile* `-o`*fmt outputfile*

where *fmt* is one of `sd`, `mae` (Maestro) or `mm` (MacroModel). Either the input or output file must be an SD file. If the input file name is specified as a dash (`-`) then input is read from standard input; likewise, if the output file name is specified as a dash (`-`) then output is written to standard output. Compressed SD and Maestro files are indicated by a filename terminating in `.gz` (or `.maegz` for Maestro files.) Compressed files can be both read and written. Files in both V2000 and V3000 format can be read and written. The format is detected automatically on

input, and V3000 format can be requested as output. V3000 format is automatically written if the structure contains more than 999 atoms.

Example:

```
$SCHRODINGER/utilities/sdconvert -n 1: -isd lig.sdf -omae lig.mae
```

1.5 Conversion To and From Mol2 Format

mol2convert

Convert structure files between Maestro and Sybyl Mol2 format. The syntax is:

```
mol2convert [-n structrange] {-imae|-imol2} inputfile
     {-omae|-omol2} outputfile
```

The input and output files must be specified.

Example:

```
mol2convert -imae myfile.mae -omol2 myfile.mol2
```

1.6 Conversion To and From SMILES: uniquesmiles

This utility generates Unique SMILES strings for the input structures, and can remove repeated strings from the output.

You can specify either a Maestro file (`.mae`) or a SMILES file (`.smi`) or both as the output files. If you specify a Maestro file, the SMILES string is stored as a string-valued property named Unique SMILES or Unique SMILES Stereo, depending on whether stereochemical information is included in the string. (The internal names for these properties are `s_canvas_Unique_SMILES` and `s_canvas_Unique_SMILES_Stereo`). The syntax is:

```
uniquesmiles [options] input-file output-files
```

2 Structure Extraction and Filtering

This section describes utilities for extracting a subset of structures from a structure file or a database. The first two utilities, `maesubset` and `sdsubset`, extract structures by index or title. The third utility, `getpdb`, extracts a structure or a chain from the PDB database.The fourth utility, `ligfilter`, extracts structures from a Maestro-formatted file based on values of properties.

2.1 maesubset

This utility extracts a subset of structures from a Maestro format input file, by index or by title. The syntax is:

```
maesubset {-n range|-range_file file} full-file > subset-file
maesubset -t {title|titlefile} full-file > subset-file
maesubset [-h] [-v]
```

The input and output files can be uncompressed (`.mae`) or compressed (`.maegz` or `.mae.gz`).

2.2 sdsubset

This utility extracts a subset of structures from an SD format input file. The input and output files can be compressed (`.sdf.gz` or `.sdfgz`) or uncompressed format (`.sdf`). If you provide both a list of structure indices or ranges and a list of titles, the resulting subset consists of structures that match both the index range and the titles. The syntax is:

```
sdsubset [-n {range|rangefile} | -t {title|titlefile}] fullfile {>|-o} subsetfile
sdsubset [-h] [-v]
```

2.3 getpdb

The `getpdb` utility extracts entire proteins or single chains from the PDB database, either as installed locally, or from the RCSB web site. The syntax is:

```
$SCHRODINGER/utilities/getpdb [options] id1 [id2 ...]
```

The *id* is in the form *PDB-code*`[:`*Chain-ID*`]`. Multiple proteins or chains can be specified in a space-separated list.

The database is located by searching the following list of locations, in order:

- `$SCHRODINGER_PDB`
- `$SCHRODINGER_THIRDPARTY`
- `$SCHRODINGER/thirdparty`
- The RCSB web site

The `-l` option restricts the search to local copies of the PDB, and does not look on the web. The `-r` option bypasses the local copies and downloads directly from the web.

If you are using a web proxy server, see Appendix C of the *Installation Guide* for instructions on setting up access.

To extract an entire protein from the PDB database, enter:

```
$SCHRODINGER/utilities/getpdb PDB-code
```

For example, the following command retrieves the PDB file for the structure 1AAA:

```
$SCHRODINGER/utilities/getpdb 1aaa
```

To extract a single chain, enter:

```
$SCHRODINGER/utilities/getpdb PDB-code:Chain-ID
```

For example, the following command extracts all residues (including HETATMs) in chain A and HETATMs with no chain name:

```
$SCHRODINGER/utilities/getpdb 1ems:A
```

Note: The *Chain-ID* variable is case sensitive, though the *PDB-code* variable is not. In this example, `1EmS:A` would also work, but `1ems:a` would produce errors. A chain ID can only be specified for PDB and FASTA formats.

2.4 ligfilter

The `ligfilter` utility filters a structure file based on properties and descriptors. Despite the name, `ligfilter` can be used to filter protein structures as well as ligand structures. It can filter on any Maestro property, a set of predefined feature counts, or counts of SMARTS patterns for functional groups. The output file reports the criteria that a structures failed to meet when it does not pass the filter.

The command syntax for `ligfilter` is:

```
ligfilter [options] input-file
ligfilter [-h] [-v]
```

where *input-file* is the file of structures to be filtered, in Maestro or SD format, uncompressed or compressed.

The syntax of a filter criterion is as follows:

```
name [ op value [ {AND|OR} op2 value2 ... ]]
```

Here, *name* is the full Maestro name of a property or descriptor, e.g. `r_qp_volume` or `i_qp_n_stars`. The conditional operator *op* must be one of the following:

```
>   >=  <   <=  ==  !=
```

and it must be surrounded by white space. If a criterion is simply *name*, then the named property is required to exist, but it may have any value. If the value for a string property contains spaces, the value must be enclosed in quotes.

Multiple conditions can be specified with the use of the `AND` and `OR` operators. For example, to specify that the property `r_user_minima` can have the values 0, 3, and 6, the filter criterion would be

```
r_user_minima = 0 OR = 3 OR = 6
```

If filter criteria are supplied in an input filter file, there must be one criterion per line. Lines that start with `#` are treated as comments. Blank lines are ignored.

The properties and feature counts that can be used are as follows:

- Molecular_formula
- Molecular_weight
- Num_aliphatic_rings
- Num_aromatic_rings
- Num_atoms
- Num_chiral_centers
- Num_heavy_atoms
- Num_heteroaromatic_rings
- Num_molecules
- Num_negative_atoms
- Num_positive_atoms
- Num_residues
- Num_rings
- Num_rotatable_bonds
- Percent_helix
- Percent_loop
- Percent_strand
- Total_charge

The list of functional groups that are defined by SMARTS patterns is extensive, and can be found in the following file:

`$SCHRODINGER/mmshare-v`*version*`/data/ligfilter_definitions.lff`

Each of the `DEFINE` lines in this file defines a functional group. The name of the group, which you can use in a filter criterion, is the next text field after the word `DEFINE`. You can copy this file and modify it to customize the definitions. When `ligfilter` is run, it looks for this file in the following locations, in the order given:

- The current working directory
- The user resources directory (`~/.schrodinger` on Linux, `%APPDATA%\Schrodinger` on Windows)
- The installation data directory, `$SCHRODINGER/mmshare-v`*version*`/data`

2.5 merge_duplicates

This script merges structures from the specified structure files. It removes duplicates based on the unique SMILES string. Before generating the SMILES, structures can be optionally desalted and neutralized. Properties for duplicate structures are merged (concatenated into a comma- separated string if a given property appears in more than one duplicate structure). This script is designed to handle tens of millions of structures. ().

The syntax is:

`merge_duplicates` [*options*] *file1* [*file2* ...]

The input file can be in Maestro, SMILES, or CSV format. The output is saved by default to a 2D SD file with structures generated from the desalted/neutralized SMILES pattern. The file name is *jobname*`-exported-`*NNN*`.sdf` (where *NNN* is an integer). This ensures that no single output file has more than 1 million structures in it, to avoid file system problems. Other valid output formats are SMILES format, *jobname*`-exported.smi` or Canvas SMILES-CSV format, *jobname*`-exported.csv` file, which can be chosen using options. The order of the structures in the output file should be treated as unpredictable.

3 Property Utilities

This section lists utilities that operate only on the properties in a structure file.

3.1 Listing of Properties:proplister

This utility lists properties in Maestro, SD, Mol2, or PDB files. The syntax is:

`proplister` [*options*] *input-files*

4 Utilities for Maestro Files

This section lists various utilities that require a Maestro file as input.

4.1 Structure Preparation: applyhtreat

Apply a hydrogen treatment (add or delete hydrogens) to one or more structures in a file. The syntax is:

`applyhtreat` *input-file output-file* [*options*] `[-t]` *treatment* `[[-a]` *ASL-expression*]

The input file and output file can be in any format that is supported by Maestro.

The `applyhtreat` utility can be used to delete or add hydrogens to one or more structures in a structure file, consistent with the hydrogen treatment option provided. A *hydrogen treatment* is a protocol that determines which atoms are to have hydrogens and lone pairs attached. Several different treatments are supplied. Each treatment is associated with a particular molecular mechanics force field, but some treatments are suitable for several force fields. See Table 2 for information on the correspondence of treatments to force fields.

This utility is used by Maestro when a hydrogen treatment is applied using the Add Hydrogens toolbar button or the Hydrogen Treatment panel. See Section 5.10 of the *Maestro User Manual* for information.

Table 2. Appropriate hydrogen treatments for Maestro-supported force fields.

Hydrogen Treatment Method	MM3	MM2	AMBER	AMBER94	MMFF	MMFFS	OPLS-AA, OPLS_200x	OPLS
All-atom with No-Lp	X	X		X	X	X	X	
All-atom with Osp3-Lp		X						
Csp3 United-atom with S-Lp[a]			X					
Csp3 United-atom with No-Lp[a]								X
All atom with S-Lp			X					
Csp2/sp3 United atom with No-Lp[a]								X

a. In a united atom representation, hydrogen atoms are incorporated into the dimensions of the heavy atom to which they are attached. That is, they are implicit.

Hydrogen addition or deletion takes place only as necessary to make the structure consistent with the selected hydrogen treatment. For example, if the specified treatment option calls for all atoms to have hydrogens and the structures in the input file already have hydrogens on all atoms, no changes are made.

4.2 Validating Maestro format: maevalidate

This utility validates the format of a Maestro file, detecting structure blocks that have incorrect syntax. The return code is nonzero if the file has incorrect syntax. The utility can be used to filter out badly formed structure blocks. The syntax is:

`maevalidate [`*options*`] `*inputfile*

The input and output files can be compressed or uncompressed; the compression is determined by the extension used.

Example output is given below. The notation `line` *n*`:`*m* means column *m* of line *n*.

```
Errors in structure 2
line 82:4 invalid property name - 'r_m' not of the form
'[irsb]_<owner>_<name>'.
line 97:4 missing ':::'
line 102 number of property names 18 != 15 property values

Error in structure 8
line 523:13 number of indexed block elements 11 != 12 declared size

Errors in structure 9
line 593:2 number of property names 2 != 3 property values
line 607:4 invalid property name - 'x_m_color' not of the form
'[irsb]_<owner>_<name>'.

Errors in structure 10
line 668:2 number of property names 2 != 1 property values
line 705:2 syntax error near 'm_bond[26'
```

4.3 Assigning Unique Names: unique_names

This program takes files in Maestro format and adjusts the numeric suffixes for structure titles and entry names to make them unique. This is useful for post-processing output from programs, such as LigPrep and MacroModel, that produce structures with identical titles and entry names. The syntax is:

`unique_names` *input-file output-file*

Note: If the structures in the input file already have unique names, these names might not be preserved if they only differ from a non-unique name by the numeric suffix.

Example:

The file `ligprep.mae` contains the following structures:

Entry Name	Title
entry	title
entry	title2
entry	title2
entry	title3

To create unique titles and entry names for these four structures, enter the following command:

```
unique_names ligprep.mae ligprep-out.mae
```

The structures in the resulting file `ligprep-out.mae` now have unique names:

Entry Name	Title
entry	title
entry1	title2
entry2	title3
entry3	title4

4.4 Protein-Ligand Interactions: ligand_interaction_diagram

This program reads a Maestro file containing a protein receptor and a ligand, and generates a diagram of the interactions between the two, as an image file. The syntax of the command is:

`ligand_interaction_diagram -i` *infile* `-o` *outfile* [*options*]

The input file can be in any 3D structure file format recognized by Maestro. The file must include a protein and one or more ligands (such as in a pose viewer file) or a ligand-receptor complex. The output is one or more image files in PNG or JPEG format, which must have the extension `.png` or `.jpg`. A numeric index starting at 1 is appended to the stem of the output file name if multiple images are generated.

5 General-Purpose Utilities

This section describes utilities that are for general use, and are not specific to Schrödinger software.

5.1 Creating and Extracting Zip Archives: ziputil

This utility can be used to create and extract zip archives with compression. It is particularly useful when built-in utilities are not capable of handling large archives, such as those that exceed the 4GB limit for normal zip format.

The command syntax for `ziputil` is

`ziputil` *action zip-file* [*filename*|*directory*]
`ziputil -h`

The use of the file name or directory argument is indicated in the descriptions. The second form of the command simply displays a usage message.

5.2 Unix Utilities for Windows

A subset of Unix commands is available on Windows in the Schrodinger Command Prompt window. These commands are in the default path in this window, so they can be used just as they are on Unix. The supported commands are:

awk	egrep	mkdir	touch
cat	env	mv	tr
chmod	find	rm	uname
cmp	grep	sed	wc
cp	gunzip	sleep	which
date	gzip	tail	whoami
diff	less	tar	xargs
dirname	ls	tee	zcat

The options and behavior are in most cases like their Unix or Linux counterparts, but in some cases the options or behavior can differ. The executables for these commands are in the `unxutils` folder of your installation.

For usage information, enter the command with the `-h` option.

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®