General Utilities

Schrödinger Suite 2012 Update 2

Schrödinger Press

General Utilities 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

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 `$SCHRODINGER/utilities` is not in your path, you should prepend the command with `$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.

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
`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 options are described in Table 1.

Table 1. Options for the structconvert command.

Option	Description
`-a`	Append to output file instead of overwriting
`-stereo=`*stereo*	Specify stereochemistry source when writing to SMILES. `none`—don't include stereochemistry `3d`—derive stereochemistry from the 3d structure (default) `annotation`—derive stereochemistry from pre-existing properties
`-n` *range*	Specify the range of structures from the input file to convert. The format of range is described above. Only valid for conversion between Maestro, SD, and Mol2 format.
`-name` *column*	Specify the column in the CSV file that contains the title. Can be given as the column index (starting at 1) or the column name. Only valid with `-icsv`.
`-smi` *column*	Specify the column in the CSV file that contains the SMILES string. Can be given as the column index (starting at 1) or the column name. Only valid with `-icsv`.

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`—see Table 2.

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 3.1.6 on page 46. 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. The options are described in Table 2.

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.

Table 2. Options for the pdbconvert command.

Option	Description
`-all_occ`	Read all alternate positions. Same as `-occ all`.
`-brief`	When writing PDB files, write only the ATOM, HETATM, CONECT, title, and pdb version records.
`-charge` *method*	Specify method for adjusting the charges on the termini. Allowed values are: `ph7_protonate_caps`—protonate termini. Termini are identified from SEQRES records if available (default). `full`—protonate termini and chain breaks. `neutral`—do not protonate termini, leave them neutral.
`-data` *datafile*	Read *datafile* for more default data, such as templates, charge information, and multiple bond distances.
`-first_occ`	Read the first of a set of alternate positions. Same as `-occ first`.
`-hex`	Use hexadecimal encoding for atom numbers greater than 99999 and for residue numbers greater than 9999.
`-histidine` *type*	Specify how to treat histidine (HIS) residues that lack hydrogens. Allowed values of *type* are: `delta` Treat histidine as HID (default). `epsilon` Treat histidine as HIE. `protonated` Treat histidine as HIP.
`-hybrid36`	Use the hybrid36 scheme for atom serial numbers. On input, integers of up to 6 digits and hexadecimal numbers are recognized on ATOM records by default. On output, the default is to use integers for less than 100 000 atoms, and hexadecimal for 100 000 atoms or more.
`-ignore_obsolete`	Ignore `OBSOLETE` records in the PDB file. Default is to fail.
`-model` *number*	Specify an input model other than the first, for PDB to Maestro conversion.
`-multbonds`	Predict multiple bonds by geometry.
`-n` *list*	Range of structures to convert from Maestro to PDB format. See page 7 for syntax. Default: convert all structures.
`-no_color`	Color atoms by type, not “error code”
`-no_dup_conect`	For PDB input, do not interpret duplicate `CONECT` records to mean multiple bonds. For PDB output, do not write duplicate `CONECT` records to represent multiple bonds.
`-no_fixelem`	Do not attempt to recognize branch codes in names like `'AC3*'` or `'NN7 '`, so that the proper element can be determined.

Table 2. Options for the pdbconvert command. (Continued)

Option	Description
`-no_geometry`	Do not predict multiple bonds by geometry. Only use `CONECT` records.
`-no_renum`	Retain atom indices from the Maestro or MacroModel file
`-no_reorder`	Retain atom indices from the Maestro or MacroModel file
`-num_models` *num*	Specify the number of models to convert from PDB to Maestro format.
`-occ [all|highest|first]`	Specify which alternate positions to read. `all` Read all alternate positions. `highest` Read the position with the highest occupancy (default). `first` Read the first of a set of alternate positions.
`-psp`	Treat Prime and PrimeX residues as standard residues for sorting. Used internally; not for general use.
`-remediated`	For PDB input, indicates that the PDB file contains remediated atom names. The `mmpdb_v3.ini` template file is used for templating. On PDB output, forces writing of REMARK 4 lines to indicate that the file contains remediated atom names.
`-reorder_by_resnum`	Write the PDB file ordered by residue number rather than by Maestro atom number.
`-reorder_by_sequence`	Write output PDB file in sequence order, i.e. from N-terminus to C-terminus. The connectivity is used to determine the sequence.
`-use_component_dict`	Use the Chemical Component Dictionary (http://www.wwpdb.org/ccd.html) when reading and writing PDB files. On reading, bonds for het residues are based on the information from the component dictionary. On writing, HET, HETNAM, and FORMUL records are written to the PDB file. The dictionary is included in mmshare.
`-warn_h`	Include hydrogens in missing atom and unrecognized atom name warnings.
`-warn_obsolete`	Print a warning if `OBSOLETE` records are encountered in the PDB file. Default is to fail.
`-write_sequence_headers`	Write SEQRES, HELIX, SHEET, TURN and SSBOND records.

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 4.

Table 3. 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.

The options are described in Table 4.

Example:

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

Table 4. Options for the sdconvert command.

Option	Description
`-a`	Append structures to the output file. Default: overwrite or create new file.
`-all`	Convert all structures (synonym for `-n 1:`). This is the default action.
`-cgch_rgroups`	Convert R# atom types to H atoms in that position and assign the atom-based integer property `i_cgch_from_atom` to the atom number of the attached atom. This option is mainly intended for CombiGlide core hopping use.
`-h`	Show usage summary
`-invert` *axes*	Specify the axes for which the coordinates are to be inverted (negated). *axes* can be composed of the characters `x`, `y`, and `z`. For example: `-invert x` negates the *x* coordinates (reflection in the *yz* plane) `-invert xyz` negates the *x*, *y* and *z* coordinates (inversion) `-invert xz` negates the *x* and *z* coordinates (180° rotation about the *y* axis)
`-n` *structs*	Specify a range of structures to convert. See page 7 for syntax. If multiple `-n` options are given, the rightmost takes precedence. Default: convert all structures.
Maestro-to-SD only:	
`-annstereo`	Derive chirality information only from Maestro stereo annotation properties.
`-pKa`	Record pK_a values from the Maestro input file in the SD output file.

Table 4. Options for the sdconvert command. (Continued)

Option	Description
`-v3`	Write output SD files in V3000 format. Default: write in V2000 format for less than 1000 atoms, write in V3000 format for 1000 atoms or more.
SD-to-Maestro only:	
`-bad`	Write structures that could not be converted to a separate file. The name of this file is the name of the input file with `-bad` added to the file stem: if the input file is *infile*`.`*ext*, the file of unconverted structures is *infile*`-bad.`*ext*. Default: discard unconverted structures.
`-noarom`	Do not convert aromatic type 4 bonds to single and double bonds, according to Maestro and OPLS conventions.
`-nolewis`	Do not fix bond orders and formal charges of nitro and carboxylate groups as appropriate for OPLS use.
`-nostereo`	Do not convert parity and bond-direction information into chirality properties, which are recorded in the structures produced.
`-notypes`	Import all properties as strings, and ignore the data type of the property.
`-rawtypes`	Assign the property type for each structure as it is read, rather than assigning it after all structures have been read. A given property in the SD file can therefore be imported as integer, real, string, or boolean for different structures.
`-skip_ambiguous`	Skip structures that contain bonds of orders 5-8.
`-title` *prop*	Define SD property *prop* as the source for assigning the Maestro title.

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. The arguments are described in Table 5.

Example:

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

Table 5. Arguments for the mol2convert command.

Argument	Description
`-bad`	Write structures that could not be converted to a file in Mol2 format. The file name is constructed from the input file name by inserting `-bad` before the `.mol2` suffix.
`-imae` *inputfile*	Input file in Maestro format
`-imol2` *inputfile*	Input file in Mol2 format
`-omae` *outputfile*	Output file in Maestro format
`-omol2` *outputfile*	Output file in Mol2 format
`-n` *structrange*	Range of structures to convert. See page 7 for syntax. Default: convert all structures.
`-noarom`	Convert aromatic atoms and bonds to nonaromatic type when converting to Mol2 format.

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*

The options are described in Table 6.

Table 6. Options for the uniquesmiles command.

Option	Description
`-dupes` *dupes*	Choose destination of duplicate SMILES strings. Allowed values are: `ignore` Do not check for duplicates (keeps all input structures) `discard` Discard duplicate strings (including conformers) `save` Write duplicates to a separate file named *outname*`-dupes.mae` or *outname*`-dupes.smi` Default: `ignore`.
`-h[elp]`	Show usage summary.
`-nostereo`	Specify that no stereochemistry should be used. Same as `-stereo none`.
`-quiet`	Run with as little output as possible.
`-stereo` *stereo*	Specify a stereochemistry option. Allowed values are: `none` Do not store stereochemical information `3d` Determine stereochemistry from 3D structure `annotation` Determine stereochemistry from properties in the input. Default: `annotation`.
`-v[ersion]`	Show the version number of this utility.
`-verbose`	Run with verbose output.

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 and fifth utilities, `propfilter` and `ligfilter`, extract 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`). The options are described in Table 7.

Table 7. Options for the maesubset command.

Option	Description
`-h`	Show usage summary
`-n` *range* | *rangefile*}	Extract structures specified by *range* or listed in the text file *rangefile*. See page 7 for syntax. The structures do not have to be specified in ascending order, and they are written out in the order specified. Default: extract all structures.
`-not`	Extract structures whose titles are *not* listed in the text file specified with the `-t` option.
`-range_file` *rangefile*	List the structures to extract in the specified file. See page 7 for syntax. This is an alternative to `-n`.
`-t` {*title*| *titlefile*}	Extract the structure whose title is specified by *title* (quoted if necessary) or the structures whose titles are listed, one per line, in the text file *titlefile*. If *title* is the name of a file, it is interpreted as *titlefile*.
`-v`	Show the version number of this utility.

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]
```

The options are described in Table 8.

Table 8. Options for the sdsubset command.

Option	Description
`-h`	Show usage summary
`-n` {*range*|*rangefile*}	Extract the structures specified by *range* or listed in the text file *rangefile*. See page 7 for syntax. The structures do not have to be specified in ascending order, and they are written out in the order specified. Default: extract all structures.
`-o` *subsetfile*	Specify the output file name. The output file extension can be `.sdf`, `.sdf.gz` or `.sdfgz`; the latter two extensions request compression of the output (with gzip). The default is to write structures to stdout, uncompressed.
`-t` {*title*|*titlefile*}	Extract the structure whose title is specified by *title* (quoted if necessary) or the structures whose titles are listed, one per line, in the text file *titlefile*. If *title* is the name of a file, it is interpreted as *titlefile*.
`-v`	Show the version number of this utility.

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 options are described in Table 9.

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

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

Table 9. Options for the getpdb command.

Option	Description
`-d`	Print debugging output
`-h[elp]`	Display the usage message and exit.
`-l`	Only search local directories for PDB files, do not search the web.
`-proxy_gui`	If proxy information is needed for web retrieval, obtain it using a GUI. Default is to use the command line.
`-r`	Only retrieve PDB files from the web, do not search local directories.
`-verbose`	Print verbose debugging output
`-v[ersion]`	Display the program version and exit

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 propfilter

This utility filters a Maestro-formatted input file based on properties. It can filter on QikProp property names or internal Maestro property names. The syntax is:

```
propfilter [options] input-file
propfilter [-h] [-n] [-v]
```

The options are described in Table 10.

Table 10. Options for the propfilter command.

Option	Description
`-e` *filter-condition*	A single filter condition. Multiple such conditions may be specified.
`-f` *filter-file*	Input file listing filter conditions to be used.
`-h`	Show the usage summary and exit.
`-n`	List the recognized property names. (The input file is not processed.)
`-o` *output-file*	Filename for output Maestro- formatted file. If no output file is specified, structures are written to the file `propfilter.mae`.
`-v`	Show the version number of this utility and exit.

Filter conditions must be specified in the form "*name op value*", or just *name*. Here, *name* is the name of a property or descriptor.

name

- For QikProp properties, the standard CSV-file header name may be used, e.g., `MW` or `#nstars` or `dip^2/V`
- The internal Maestro property names may also be used, e.g., `r_qp_mol_wt`, or `i_qp_n_stars` or `r_rp_d2ov`.

op

The conditional operator *op* must be one of the following:

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

and it must be surrounded by white space. The ~ and !~ operators perform pattern matching using regular expressions.

If a condition is simply *name*, then the named property is required to exist, but it may have any value.

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

2.5 ligfilter

The `ligfilter` utility filters a structure file based on properties and descriptors. It can filter on any Maestro property, a set of predefined feature counts, or counts of SMARTS patterns for functional groups. This utility supersedes `propfilter`, and should be used in preference to it. Input filter files from `propfilter` can be read by `ligfilter`. The output file reports the criteria that a ligand 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 options are described in Table 11.

Table 11. Options for the ligfilter command.

Option	Description
`-addprops`	Add counts as Maestro properties to the output structure file.
`-any`	Require match to at least one criterion instead of all criteria. Using this option is implicitly inserting an OR between each criterion, instead of an AND.
`-asl` *expression*	ASL expression to match. At least one atom in any structure must match the ASL expression for the structure to pass the filter.
`-asl_file` *filename*	Name of a file containing one or more ASL expressions to match.
`-doc`	Display extended documentation.
`-e` *filter-condition*	A single filter criterion. Multiple such criteria may be specified.
`-f` *filter-file*	Input file listing filter criteria to be used. Each criterion should be listed on a single line. Can also include definitions of functional groups.
`-h[elp]`	Show usage summary.
`-invert`	Discard matching molecules instead of retaining them.
`-j` *jobname*	Specify job name. The default is the name of the input file minus the extension.
`-n` *nonmatch-file*	Filename for output of molecules that do not match the conditions.

Table 11. Options for the ligfilter command. (Continued)

Option	Description
`-noout`	Do not write an output structure file. This option is useful for obtaining counts of structures from the log file without generating the structure file.
`-o` *output-file*	Filename for filtered structure output. The default is *jobname*`-out.`*ext*, where *ext* is the extension of the input file.
`-v`	Show the version number of this utility.

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`

2.6 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.

The options are listed in Table 12.

Table 12. Options for the merge_duplicates command.

Option	Description
`-add`	Only add the specified files to the stored database. This option allows you to add files one by one, and export them to a file later.
`-codefield` *codefield*	Name of the property that is used to store the generated compound code. This code is an assigned integer value. By default, no compound code is generated.
`-codeprefix` *codeprefix*	Prefix that will be prepended to the compound code. By default, no prefix is used when generating a compound code.
`-csv`	Export structures to CSV format instead of 2D SD format
`-desalt`	Desalt the structures before merging. This action removes all molecules except the ligand.
`-export`	Export stored structures
`-h[elp]`	Show the usage message and exit.
`-j` *jobname*	Job name. Default: `merge`.
`-JOBID`	Run the script under job control (forced when -HOST is used)
`-max_per_file` *number*	When exporting to SD, split the output so that at most this many structures are in each file (disabled by default).
`-neutralize`	Neutralize the structures before merging, by protonating or deprotonating.
`-no_cleanup`	Do not remove intermediate files.
`-nomerge`	Do not merge properties. If a given property is found in more than one duplicate structure; keep just one value. By default, duplicate values are concatenated using commas.
`-OVERWRITE | -overwrite`	Overwrite previous results
`-requireprops`	Skip structures that do not have any properties
`-RESTART | -restart`	Restart the previous job
`-smi[les]`	Export structures to SMILES (`.smi`) format instead of 2D SD format. Only the title is exported.
`-smilesfield` *smilesfield*	Name of the property that is used to store the SMILES string on export to an SD file. If omitted, SMILES strings are not exported.
`-summary`	Print summary only
`-v[ersion]`	Show the program version and exit.

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*

The options are described in Table 13.

Table 13. Options for the proplister command.

Option	Description
`-a`	Report *all* the properties found in the input files.
`-c`	Write properties to a CSV file. The default delimiter is a comma.
`-d` *delimiter*	Specify the delimiter to be used in the CSV file.
`-h`	Show usage summary.
`-l`	List the names of all the properties found in the input files.
`-n`	List the property names for which user-friendly names are known. (Input files are ignored).
`-noheader`	Report only the properties, do not write out header lines.
`-o` *outfile*	Write output to file *outfile* instead of stdout.
`-p` *property-names*	Name of property to report. The property name can be the full name, e.g. `s_m_title`, or the short name, e.g. `title`. If the short name is used, underscores must be replaced by spaces and the name must be enclosed in quotes. If more than one property with unrelated names is needed, use multiple `-p` options, e.g., `-p title -p MW`. You can specify multiple properties by using the wildcard characters `?` for a single character and `*` for any number of characters. If you use wildcard characters, you must enclose the names in single quotes so that they are not interpreted by the shell.
`-t`	Use tab as the delimiter in the CSV file.
`-u`	Print property names in Maestro-visible format, by removing the first two fields and replacing underscores with blanks. For example, the property `s_family_My_Prop` is written as `My Prop`.
`-v`	Show the version number of this utility.

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 options are described in Table 14.

Table 14. Options for the applyhtreat command.

Option	Description
`-a[sl]` *asl*	Specify a range of atoms to which the treatment is to be applied, as an ASL expression. The expression must be enclosed in quotes. The `-a` is optional if the ASL expression follows the treatment.
`-h[elp]`	Show usage message and exit.
`-s | -use_stereo`	Adjust atom numbers in chirality strings to reflect the addition of hydrogen atoms.

`-t[reatment]` *name*	Specify the hydrogen treatment name. The treatment name must be enclosed in quotes. The `-t` is optional if the treatment is the first non-option argument. Allowed treatment names are:
	`All-atom with Osp3/Nsp3-Lp`
	`All-atom with No-Lp`
	`Csp3 United-atom with S-Lp`
	`Csp3 United-atom with No-Lp`
	`All-atom with S-Lp`
	`C sp2/sp3 United-atom with No-Lp`
	`C sp2/sp3, N,O,S United-atom with No-Lp`
	The default is `All-atom with No-Lp`.

`-v[ersion]`	Show the version number of this utility and exit.

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 15 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 15. 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. The options are described in Table 16.

Table 16. Options for the maevalidate command.

Option	Description
`-og` <*filename*>	Output file to write valid (good) structures.
`-ob` <*filename*>	Output file to write invalid (bad) structures.
`-r`	Use line numbers relative to the beginning of a structure block in error messages, rather than absolute line numbers.
`-q`	Suppress error messages. Useful if no file output is needed, only the return code, which is nonzero if the file has errors.

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. The options are given in Table 17.

Table 17. Options for the ligand_interaction_diagram command.

Option	Description
`-h[elp]|--help`	Display usage message and exit.
`-ligandASL` *ASL*	ASL expression used to identify the ligand. The default is the same as for ligand detection in Maestro, which includes molecules with 5–130 atoms and excludes certain ions and common small molecules.
`-residuesASL` *ASL*	ASL expression used to define the protein residues to display. The default is the residues within 4 Å of the ligand.
`-showLegend`	Include a legend in the diagram for the features that are shown.
`-showFullLegend`	Include a legend in the diagram that includes all possible features.
`-showTitle`	Include the title of the ligand at the top of the diagram.
`-rotationMatrix16` *list*	Use the following 16 numbers as the rotation matrix for the orientation of the structure.
`-view` *filename*	Specify a Maestro `.views` file for the orientation of the structure.

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 actions for ziputil are described in Table 18. 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.

Table 18. Actions for the ziputil utility.

Action	Description
`--extract`	Extract the specified file from the zip file, if it exists.
`--create`	Create the zip file and add the specified file.
`--createdir`	Create the zip file and add the specified directory (with its contents).
`--create64`	Create a Zip64 format zip file and add the specified file.
`--createdir64`	Create a Zip64 format zip file and add the specified directory (with its contents).
`--append`	Append the specified file to the archive or replace it if it already exists in the archive.
`--appenddir`	Appends the specified directory (with its contents) to the archive.
`--unzip`	Uncompress the entire archive.
`--remove`	Remove the specified file.
`--info`	Display some information about the archive.
`--list`	List the files in the archive.

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

Dynamostraβe 13
D-68165 Mannheim
Germany

Zeppelinstraβe 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®