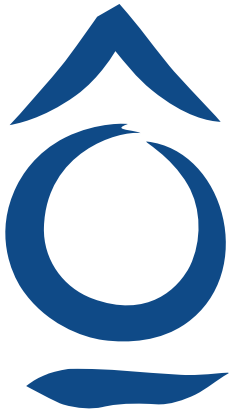


# OpenEye SCIENTIFIC

# QUACPAC

***Release 1.7.0.2***

**OpenEye Scientific Software, Inc.**

October 04, 2016

CONTENTS

CHAPTER

ONE

# FRONT MATTER

CHAPTER

TWO

# INSTALLATION AND PLATFORM NOTES

## 2.1 Licenses

To run QUACPAC and the associated utilities you will need to obtain a license file for QUACPAC from OpenEye Scientific Software (sales@eyesopen.com). The license file should be in a file pointed to by the `OE_LICENSE` environment variable.

## 2.2 Installation

### 2.2.1 Linux

Linux distributions are provided as a gzipped tarball of the distribution tree described below. Installation is performed by simply untarring the file in the desired location. The top-level directory in the tarball is named `openeye`. Distributions for different Linux variants can be installed into the same location, allowing multiple Linux versions to be run from a single shared directory.

To ensure that the installed applications can be called from the command line, be sure to add the full path of the `openeye/bin` subdirectory to the `PATH` environment variable. For instance, if the distribution was installed into `/usr/local/openeye`, the `PATH` environment variable should contain: `/usr/local/openeye/bin`.

Under the top-level `openeye` directory are the following subdirectories:

**arch** This directory contains the collection of platform specific subdirectories. Each subdirectory contains the actual installed executables and support libraries for the associated platform. In the platform specific subdirectory there will be a subdirectory for each application. Within that will be another subdirectory for each version of that application.

**bin** This directory contains a startup script for each application that has been installed. This script determines, at run-time, what the current platform is and then calls the appropriate executable in the `arch`. This script enables the easy co-existence of multiple platforms and versions of any OpenEye application in the same distribution tree.

**data** This directory contains all of the associated data for the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

**docs** This directory contains all of the documentation associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

**examples** This directory contains all of the examples associated with the installed applications. There will be a subdirectory for each installed application and within that subdirectory there will be another subdirectory for each specific version of that application.

The startup script discussed in the section on the `bin` directory above will have the same name as the installed executable with which it is associated. When the script is called, it will attempt to determine the current platform and run the appropriate executable if installed. If an appropriate executable cannot be found, the script will report that information, as well as a list of the currently installed platforms. The auto-detection can be overridden by setting one of two environment variables:

- `OE_ARCH` can be used to specify a colon separated list of compatible distributions for the current platform such as:

  ```
  redhat-RHEL6-x64:redhat-RHEL5-x64
  ```

  Specification of this environment variable overrides the auto-detection process, if it is present. If none of the compatible distributions listed are found, the script will fall back to the auto-detection process.

- `APPNAME_OE_ARCH` can be used to specify a colon separated list of compatible distributions for a specific application (as specified by changing the **APPNAME** text in the environment variable name) just like `OE_ARCH` as detailed above.

  Specification of this environment variable overrides the `OE_ARCH` environment variable as well as the auto-detection process. If none of the compatible distributions listed are found, the script will fall back to the `OE_ARCH` list first and then to the auto-detection process.

  Specifying this variable provides a simple way to customize the behavior for individual applications on non-standard platforms.

The startup script also supports a few commandline arguments including:

| | |
|---|---|
| **-path** | Specifying this argument will output the full path of the executable to be run. The executable will not be started if this argument is present. |
| **-print_arch** | Specifying this argument will output the details of the current platform as detected by the script as well as which platform-version of the executable is being run. The executable will be started if this argument is present. |
| **-use_version** | Specifying this argument followed by a specific version number allows the user to control which released version of the executable to run. |

### 2.2.2 Windows

Windows distributions are provided as a standard EXE installer. For installation double click the executable and follow the installation instructions. By default, OpenEye applications will install into the `C:\Program Files (x86)` directory (for 32-bit applications) or `C:\Program Files` (for 64-bit applications).

---

**Note:** Because memory is limited for 32-bit applications, to avoid crashes due to running out of available memory, memory-intensive tasks are best performed using 64-bit applications.

---

Under the application directory (`C:\Program Files\Application Name`) there are subdirectories for:

**bin** This directory contains the application executable.

**data** This directory contains all of the associated data for the installed applications.

**docs** This directory contains all of the documentation associated with the installed applications.

**examples** This directory contains all of the examples associated with the installed applications.

An OpenEye group with an application specific subgroup will be added to the *Start* menu. The application specific subgroup will contain links to the documentation, the uninstaller, and, for some applications, a Windows command shell with `PATH` settings already defined to allow the user to simply type the executable name at the prompt without

concern for where the executable is actually installed. Links are also included to add and remove the installed location to the user's default path.

For graphical applications, a link to the application will be created on the desktop as well as in the application specific subgroup of the *Start* menu.

### 2.2.3 Mac OS X

Mac OS X distributions are provided as a *dmg* disk image. For installation, double click the .dmg file to open it, and drag the application to the Applications folder.

A folder containing documentation and example data is included in the disk image (Right click, `Show Package Contents`). Under the top level `Contents` folder there are subdirectories for:

**data** This directory contains all of the associated data for the installed applications.

**docs** This directory contains all of the documentation associated with the installed applications.

**MacOS/bin** This directory contains the application executable.

The documentation and example data can be copied to any convenient location. Graphical applications have built-in documentation, which is available from the application's Help menu.

For command-line only tools, an application named "Install Command Line Support" can be run from the .dmg file, and this will allow the user to add the application's location to the user's `PATH` environment variable. Command-line applications can also be run from the Applications folder, in which case they will open a terminal window with a properly configured environment.

## 2.3 Uninstallation

### 2.3.1 Linux

To uninstall a single distribution of a product the relevant subdirectories for that product and version simply need to be deleted from within the following directories:

**arch** In the `openeye/arch` directory is a platform specific subdirectory. Within this are directories for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled. For example, to delete or uninstall v1.0.0 of a product, delete the folder "<product_name>/1.0.0".

**data** In the `openeye/data` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

**docs** In the `openeye/docs` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

**examples** In the `openeye/examples` directory is a subdirectory for each installed product and within those are subdirectories for each version of the product. Delete the subdirectory for the version which is to be uninstalled.

### 2.3.2 Windows

Installation of an OpenEye product on Windows causes an OpenEye group with an application specific subgroup to be added to the *Start* menu. One of the items in the application specific subgroup is a link to the uninstaller. Clicking

on the uninstaller initiates a wizard which guides the user through uninstallation.

For graphical applications, uninstallation also removes the application's link from the desktop and *Start* menu.

### 2.3.3 Mac OS X

To uninstall a single distribution of an application simply drag the application from the Application folder to the Trash.

Some applications may, at the user's request, install symbolic links in the /usr/local/bin directory, and modify the PATH variable in the .openeyerc.* file(s) in the user's home directory. The symbolic links may be safely deleted after uninstallation, and the .openeyerc.* file(s) can be edited, if desired, to remove obsolete entries.

CHAPTER

THREE

# QUACPAC THEORY AND USAGE

## 3.1 Introduction to QUACPAC

The chemistry of molecular interactions is a matter of shape and electrostatics, but doing electrostatics poorly is worse than doing none at all; accurate charges are required. Even the best charge models are useless if protonation states are wrong. QUACPAC attempts to offer everything necessary to do charges correctly. It includes pKa and tautomer enumeration in order to get correct protonation states, partial charges using multiple models that cover a range of speed and accuracy, and electrostatic potential map construction and storage.

## 3.2 *tautomers* - Enumeration and Canonicalization

OpenEye Scientific Software's *tautomers* program is used for canonicalizing and/or enumerating the tautomeric forms of a small molecule. Canonicalization converts any of the tautomeric forms of a given molecule into a single unique representation. This is useful for database registration where alternate representations of tautomeric compounds often leads to duplicate entries in a database.

Some effort is made by the *tautomers* program to direct the "canonical" representation to be a physiologically preferred form. However, there are no guarantees the tautomer selected is indeed the lowest energy and, indeed, solvent effects, etc., preclude there being a single "best" form of a tautomer. Fortunately, this is not necessary for database work.

*tautomers* is not a conformer generation program and will not create coordinates for molecules that are read in with no coordinates. When used on molecules with three-dimensional coordinates, tautomers attempts to place hydrogens in a reasonable manner. However, tautomers does not modify the heavy-atom coordinates of the molecule. In cases where the change in tautomer-state dictates a change in conformation, one will need to use a conformer-generation tool (such as OMEGA) to generate reasonable conformations for the output from tautomers. We recommend that in the preparation of small-molecules for study, charge-state and tautomer enumeration be performed before conformer generation.

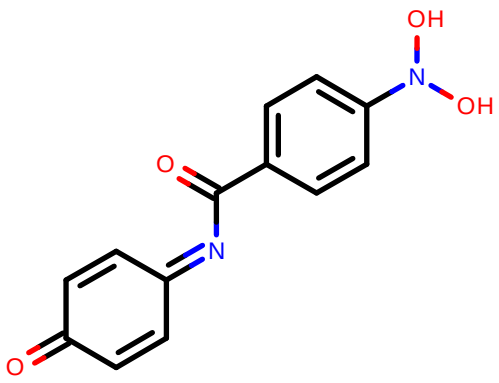
### 3.2.1 Reasonable Ranking

By default, *tautomers* will return the canonical tautomer as the first molecule in the iterator. However, if -reasonable is set to `true`, *tautomers* will attempt to provide a low-energy, medicinally-relevant "reasonable" tautomer. Since it is not feasible on an informatics time-scale to correctly predict tautomer ratios, the "reasonable" tautomer is a good trade-off for generating a tautomeric form that is suitable for depiction for chemists.

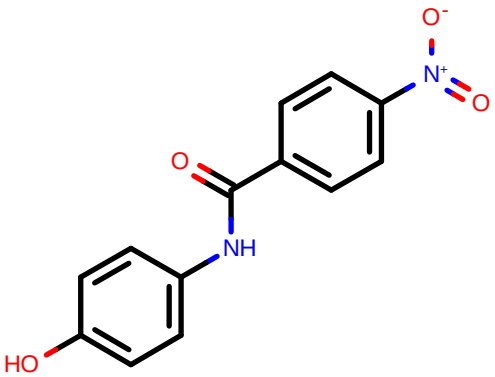
The following depictions are a useful guide to knowing what tautomers are favored as "reasonable":

- Conversion of carboxylates to diols and nitros to di-hydroxy amines is very unfavorable.

disfavored

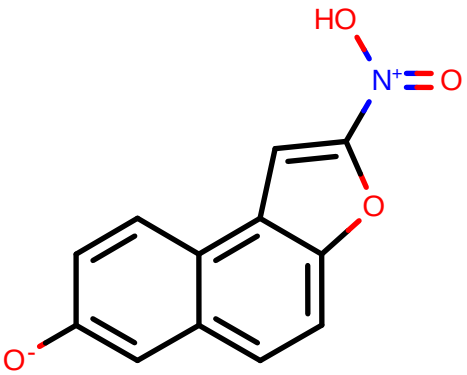



favored

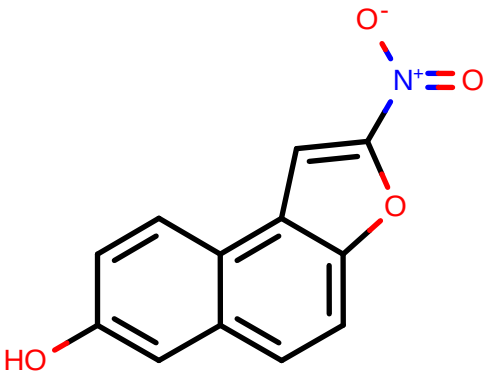



- Generation of unnecessary, non-dative, formal charges is unfavorable.

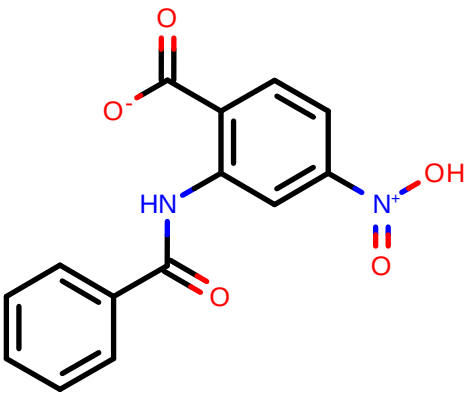
disfavored



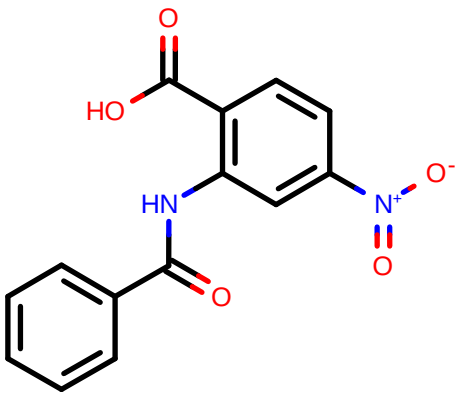

favored



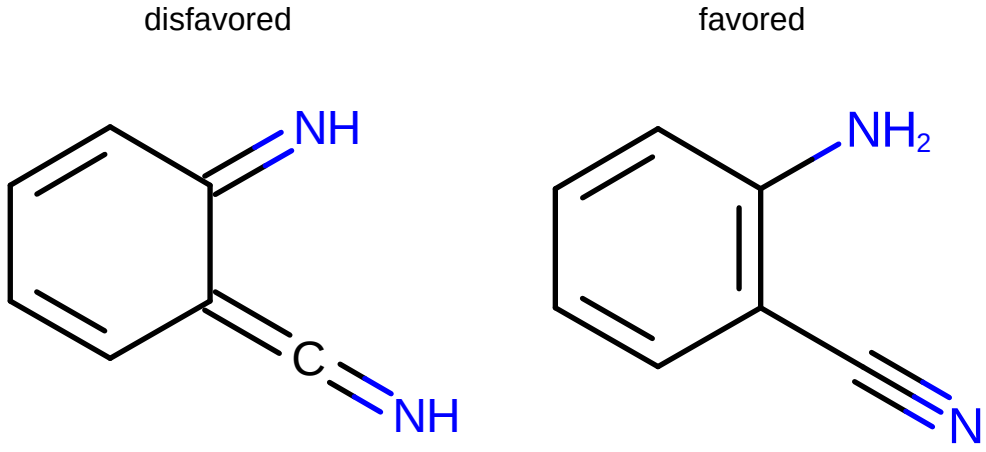

disfavored



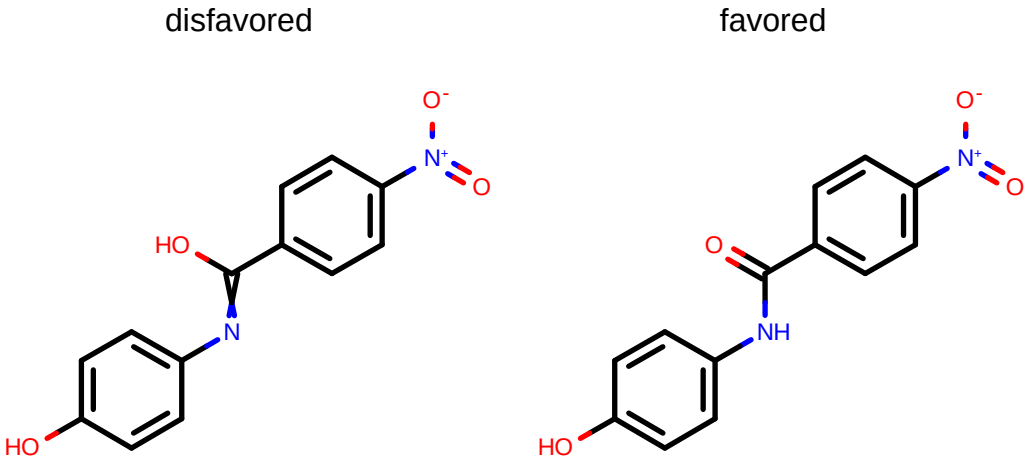

favored



- Exocyclic bonds adjacent to aromatic rings are accounted for.




- Priority is given to aliphatic double bond positions.




## 3.3 *tautomers* Usage

### 3.3.1 Required Parameters

**-in**

File containing one or more molecules for which tautomers will be generated. An input file is required, but the *-in* flag is optional. The first parameter listed with no flag will be automatically mapped to the input file. Unflagged parameters must occur last in the parameter list. Any OpenEye supported molecule format can be used for input.

### Optional parameters

**-can**
By default, output the OpenEye canonical SMILES for each tautomer. When -can has a value of false, tautomers writes arbitrary SMILES where the order of the atoms in each tautomer is the same for a given input molecule, which often makes it easier to see the differences between tautomers when reading the SMILES. [default = true]

**-ch3**
Allows carbon atoms that are close by appropriate heteroatoms to change hybridization state. This permits structures such as cyclohexa-2,4-dien-1-one to be considered a tautomer of phenol, and other examples of keto-enol tautomerism. Unfortunately, the *-ch3* flag may cause the calculation time to increase by many orders of magnitude for some molecules. [default = false]

**-count**
Output just the number of tautomers per compound rather than listing the tautomers. This option writes to a text file or stdout. [default = false]

**-kekule**
Output is in Kekule format for .smi, .ism. and .can. Disabling aromaticity often makes it easier to understand the differences between tautomers. [default = false]

**-level**
Manually set the acceptable approximate tautomer energy level to use in enumeration. This takes an integer value between zero and seven inclusive, where zero corresponds to lowest energy states and seven corresponds to the highest energy state. By default, the tautomers program enumerates all tautomers in the lowest non-empty low energy state; first trying level zero and if no tautomers are found increasing to one, then two and so on. [default = 0]

**-max**
Specify a maximum number of tautomers to enumerate for a single input structure. Over 99% of compounds require less than 100 tautomers, indeed most organic compounds gave only a single tautomer, however some pathological dyes and chromophores may individually have nearly a million possible tautomeric forms. The current default is a limit of 1000 tautomers per input structure. [default = 1000]

**-out**
Output filename, where the file extension indicates the output format. All OpenEye supported molecule formats are allowed but .smi and .ism are recommended. The default setting will print .smi format to standard out.

**-param**
The argument for this flag is the name of a file containing control parameters. The control parameter file acts to either replace or augment the command line interface. All parameters necessary for program execution may be provided in the control parameter file, although any command given explicitly on the command line will supersede options found in the parameter file. *tautomers* generates a new parameter file containing the full set of execution parameters upon every execution. The name of the parameter file written by *tautomers* is created by combining the prefix base name with the *.parm* extension.

**-paramfile**
The filename for the output parameter file can be set with this flag and it overrides -prefix flag for the name of parameter file.

**-prefix**
The argument for this flag defines the prefix to be used for parameter and log files. The parameter and log files will be written to what follows this flag plus the extension *.param* or *.log*. [default = tautomers]

**-reasonable**
Only for those who want a unique tautomer which is reasonable looking. The program will output the most aromatic tautomer of the first 64 attempted. [default = false]

**-savestereo**
This allows for any atom or bond with stereochemistry set to not take part in tautomerization. Without this setting it is possible for stereochemistry to removed during tautomerization. [default = false]

**-uniq**
Run in canonicalization mode, where for each molecule in the input file, a single "canonical" tautomer is written to the output file. By default, *tautomers* runs in enumeration mode. [default = false]

**-maxtime**
This flag limits the amount of time (in seconds) spent generating tautomer for each molecule. [default = 60.0]

**-warts**
Add wart number with _ to each tautomer. [default = true]

### 3.3.2 Example Executions

The following section shows several common command-line executions of *tautomers*. Each example is followed by an explanation of what the program will do.

Consider the following input file, *guanine.smi*, that contains just the following line: *c1[nH]c2c(=O)[nH]c(nc2n1)N*

```
prompt> tautomers guanine.smi output.smi
```

Which should write the following 15 structures to the file *output.smi*.

```
c1[nH]c2c(n1)[nH]c(nc2=O)N output_1_1
c1[nH]c2c(=O)[nH]c(nc2n1)N output_1_2
c1[nH]c2c(n1)c(=O)nc([nH]2)N output_1_3
c1[nH]c2c(n1)c(=O)[nH]c(n2)N output_1_4
c1[nH]c2c(n1)[nH]c(=N)[nH]c2=O output_1_5
c1[nH]c2c(n1)c(=O)[nH]c(=N)[nH]2 output_1_6
c1[nH]c2c(n1)nc(nc2O)N output_1_7
c1nc-2c(nc([nH]c2n1)N)O output_1_8
c1[nH]c2c(n1)c(nc(n2)N)O output_1_9
c1nc-2c([nH]c(nc2n1)N)O output_1_10
c1[nH]c2c(n1)[nH]c(=N)nc2O output_1_11
c1[nH]c2c([nH]c(=N)nc2n1)O output_1_12
c1[nH]c2c(n1)c(nc(=N)[nH]2)O output_1_13
c1nc-2c([nH]c(=N)[nH]c2n1)O output_1_14
c1[nH]c2c(n1)c([nH]c(=N)n2)O output_1_15
```

Please note that if *tautomers* fails to find tautomers for a molecule, it will output the molecule to a fail file. Name of the fail file is either 'tautomers.fail' or 'prefix.fail' if prefix is specified by *-prefix* flag.

## 3.4 *pkatyper* - Ligand pKa

### 3.4.1 Introduction

Assessment of ligand pKas can be broken into two phases. The first phase is enumeration of the protonation states of interest, and the second phase is assigning a pKa value to each of these states. An intermediate phase of assigning microscopic pKas to each of the atomic-deprotonations may also be considered.

It is common in the course of modeling small-molecules to explore the conformational ensemble of the small molecule. Often structures as high as 5-8 kcal/mol above the aqueous ground-state can be important to biological processes. It is also appropriate to enumerate a protonation-state ensemble of the small molecule.

Similar to *tautomers*, OpenEye has a solution for enumerating reasonable protonation states, but not for assessing the energetics of the state (e.g. assigning a pKa value). OpenEye's solution for pKa enumeration seeks to enumerate all of the pKa states that fall roughly in the pH range of 2-14 in aqueous solvent. This range of pKa values generates an ensemble that includes the ground-state plus all charge states within 8 kcals/mol $\Delta G$. This value was chosen to correspond to the similar range that is often used for generating conformational ensembles of small molecules.

### 3.4.2 Theory

*pkatyper* enumerates charge states based on primary, secondary and tertiary atom types of each atom in a molecule. The primary atom type is based on the atom's group and its valence. The primary atom-type defines the atom's basic propensity to support a formal charge. The secondary atom-type is defined by the atom-type of the neighbors for each atom. These secondary atom-types, such as aromaticity, alpha-beta unsaturation, or electronegative-groups, modulate each atom's basic propensity to support formal charges. The tertiary atom-types assess the effects of nearby formal charges on a given atom's formal charge. The combination of the primary, secondary and tertiary atom-types determine which formal charge states are allowed for each atom in a molecule. The primary and secondary atom-types are determined once, while the tertiary atom-types are determined as part of the enumeration process.

*pkatyper* is a rudimentary approach to pKa prediction. While pkatyper is not suited for prediction of absolute pKas, it is quite amenable to enumeration of all reasonable charge states of a very wide variety of small-molecule chemistries.

*pkatyper* is not a conformer generation program and will not create coordinates for molecules that are read in without coordinates. When used on molecules with three-dimensional coordinates, *pkatyper* attempts to place new hydrogens in a reasonable manner. However, *pkatyper* does not modify the heavy-atom coordinates of the molecule. In cases where the change in protonation-state dictates a change in conformation, one will need to use a conformer-generation tool (such as OMEGA) to generate reasonable conformations for the output from *pkatyper*. We recommend that in preparation of small-molecules for study, charge-state and tautomer enumeration be performed before conformer generation.

In the future, OpenEye will release a product which assigns a pKa value to each of the enumerated states.

**Warning:** *pkatyper* generates input suitable for higher level calculations.

## 3.5 *pkatyper* Usage

### 3.5.1 Required Parameters

**`-in`**
: Input file of molecules. An input file is required, but the *-in* flag is optional. The first parameter listed with no flag will be automatically mapped to the input file. Unflagged parameters must occur last in the parameter list.

#### Optional parameters

**`-count`**
: Only count the number of pKa states rather than enumerating each of the states. If the *-count* flag is specified true, then the output file will contain the name of each compound followed by the number states of that molecule. This options writes to a text file or stdout. [default = false]

**`-max`**
: This integer parameter is the maximum number of pKa states which will be enumerated for any single molecule. If a value of zero is passed to this parameter, no limit will be set. [default = 100]

**-out**
Output file of molecules. Both the output file and the *-out* flag are optional. The second parameter listed with no flag will be automatically mapped to the output file. Unflagged parameters must occur last in the parameter list. If no output is specified at all, output will be written to *std::out* in SMILES format. Any OpenEye supported molecule format can be used for output. If the *-count* flag is specified true, then molecular format is no longer relevant to the output file.

**-param**
The argument for this flag is the name of a file containing control parameters. The control parameter file acts to either replace or augment the command line interface. All parameters necessary for program execution may be provided in the control parameter file, although any command given explicitly on the command line will supersede options found in the parameter file. *pkatyper* generates a new parameter file containing the full set of execution parameters upon every execution. The name of the parameter file written by *pkatyper* is created by combining the prefix base name with the *.parm* extension.

**-paramfile**
The filename for the output parameter file can be set with this flag and it overrides -prefix flag for the name of parameter file.

**-prefix**
The argument for this flag defines the prefix to be used for parameter and log files. The parameter and log files will be written to what follows this flag plus the extension *.param* or *.log*. [default = pkatyper]

### 3.5.2 Example Commands

The following section shows several common command-line executions of *pkatyper*. Each example is followed by an explanation of what the program will do.

```
prompt> pkatyper drugs.sdf enumerated_drugs.smi
```

Reads the file *drugs.sdf* in SD format, enumerates the pKa states of each molecule and writes them into the file *enumerated_drugs.smi*. Molecules with only one identified pKa state are passed through to the output file.

```
prompt> pkatyper -in drugs.sdf -out enumerated_drugs.smi
```

This command generates the exact same behavior as described above.

```
prompt> pkatyper -count drugs.sdf drugCounts
```

This command reads each of the molecules in *drugs.sdf* and counts the number of pKa states. For each molecule, a single line is added to the *drugCounts* output file that contains the molecule title and the number of states.

## 3.6 *fixpka* - pKa Normalization

*fixpka* has a rule-based system to set the ionization state of input molecules. If pKa normalization is turned on, the molecule is set to its most energetically favorable ionization state for `pH=7.4`. The rule-based nature of this calculation allows it to be very fast. Further, despite being rule-based, this approach takes into account many secondary charge interactions.

While more advanced levels of theory can be found for predicting ionization states, this method is very well suited to virtual-screening database preparation. However, *fixpka* may not be appropriate for hit-to-lead or lead optimization.

## 3.7 *fixpka* Usage

### 3.7.1 Required Parameters

**-in**
: File containing one or more molecules from which pKa normalization will be done. An input file is required, but the *-in* flag is optional. The first parameter listed with no flag will be automatically mapped to the input file. Unflagged parameters must occur last in the parameter list. Any OpenEye supported molecule format can be used for input.

**-out**
: File for writing output. An output file is required, but the *-out* flag is optional. The second parameter listed with no flag will be automatically mapped to the input file. Unflagged parameters must occur last in the parameter list. Any OpenEye supported molecule format can be used for output but *.smi* or *.ism* is recommended.

#### Optional parameters

**-ionize**
: Specifies ionization environment, either neutral pH or un-ionize, to remove all ions possible. Legal values are neutral, 7.4, un-ionize, and unionize.

### 3.7.2 Example Commands

The following section shows a common command-line execution of *fixpka*.

```
prompt> fixpka drugs.sdf fixpka_drugs.smi
```

Reads the file *drugs.sdf* in SD format, sets the pKa state of each molecule and writes them into the file *fixpka_drugs.smi*.

## 3.8 *molcharge* - Partial Charges

### 3.8.1 Introduction

The assignment of appropriate atomic partial charges, both to small molecule ligands and to biopolymers (such as proteins and nucleic acids) is essential to getting meaningful results from any electrostatics calculation.

A molecule may be considered a collection of atomic nuclei and the electrons that surround them. The number of protons in each nucleus defines its atomic number/element. If the number of electrons exactly matches the number of protons in these nuclei, the molecule is neutral and has no net charge. If there are more electrons than protons, the molecule has a net negative charge, and if there are less, the molecule has a net positive charge.

It is both the atomic nuclei and the net charge that define the identity of a molecule. Indeed, this is a representation common to quantum chemistry. Adding or removing electrons (or atoms) from a molecule produces a different molecule.

In the discrete world of cheminformatics, valence bond theory allows the electrons present in a system to be represented in terms of bonds with formal bond orders, and formal charges assigned to particular atoms. The sum of the formal charges is equal to the net charge on the molecule, but which atoms are assigned which formal charges can be to some extent arbitrary due to resonance delocalization. In such cases the same molecule may be represented by similar connection tables, but with formal charges assigned to different sets of atoms.

For example, guanidinium may be expressed as either N[C+](N)N with the formal charge assigned to the carbon, or as [NH2+]=C(N)N with the formal charge assigned arbitrarily to one of the otherwise equivalent nitrogens. A similar example is a thiocarboxylate group, where either C(=O)[S-] or C(=S)[O-] are both equally appropriate representations of the same chemical functionality.

A zwitterion is an electrically neutral molecule that is represented as containing atoms with positive formal charge as well as atoms with negative formal charge.

Perhaps the most important fact to appreciate when considering formal charges on atoms is that they are all artificial constructs by chemists to accommodate a particular chemical model. A figment of a chemist's fevered imagination. Like valence bond theory, they are an exceptionally useful and powerful discretized model of the universe. But as with any model of reality, it has its limitations. Formal charges, for all their numerous benefits to mankind, unfortunately, are not localized on an atom.

The limitations of describing formal charges with valence bond theory is apparent even within cheminformatics. Sydnones, for example, are a class of heterocyclic compound that cannot be written using normal covalent bonds without introducing and arbitrarily assigning both positive and negative charges. Similarly, in inorganic chemistry, the ditechnetium cation, $Te_2^{+5}$, causes similar problems where the +5 formal charge cannot be assigned to both technetium atoms without breaking symmetry.

A better model, or approximation, of the wave function describing the distribution of electron density around a molecule is the use of atomic partial charges. A partial charge is a floating-point value assigned to each atomic center intended to model the distribution of electrons over a molecule.

Atomic partial charges are yet another approximation, much like the formal charges described above. However, partial charges provide a much better model to describe the electric field, dipole moment and other observable properties of a molecule.

A common limitation of the use of partial charges is the assumption that they are conformationally invariant. Unfortunately, the distribution of electrons around a molecule depends upon the spatial configuration of its nuclei. Some partial charge assignment algorithms, such as the method of Goddard and Rappé, consider these conformational effects, whilst others that are based on quantum mechanics, such as the RESP and AM1BCC methods of Bayly et al., go to great lengths to eliminate conformational effects, for example, by restraining and symmetrizing symmetric atom positions. This is necessary in order to be able to properly handle multiple conformations and changes in geometry (e.g. geometry optimization) with a single set of atomic charges.

### 3.8.2 Theory

#### Marsili-Gasteiger Partial Charges

Marsili-Gasteiger partial charges are assigned using a two stage algorithm. In the first stage, seed charges are assigned to each atom in the molecule. For example, carboxylate oxygens are each assigned the value -0.5. During the second stage, these initial charges are then shared across bonds, moving a certain amount of charge from one atom to another. The partial charge moved and its direction is determined by difference in electronegativities of the atoms on each end of the bond. The relaxation algorithm is then iterated several times (by default eight passes), attenuating the charge moved with each iteration. OpenEye does not recommend use of this charge model for intermolecular interactions; it was never intended for this purpose. The author of the method (Johann Gasteiger) developed it to compare relative reactivity of related organic chemical functional groups within different molecular contexts. Here it is included for comparison purposes.

#### MMFF94 Partial Charges

The partial charges used by the MMFF94 and MMFF94s force fields are assigned using a four stage algorithm. In the first stage, each atom of the molecule is assigned an MMFF94 atom type. In the second stage, an initial seed partial charge is assigned to each atom based upon its atom type. For a few atom types, the initial partial charge

also depends upon the local environment. In the third stage, the initial charges assigned to aromatic rings are shared between all atoms of the aromatic ring. Finally, in the fourth stage, a table of bond charge increments (BCI) is used to move charges across bonds based upon the bond type of the bond (single, double, triple) and the atom types of the atoms at each end. Developed for the electrostatic interactions within the above-mentioned force fields, they are the appropriate charges to use with these force fields most notably for intramolecular interactions of pharmaceutical and bio-organic small molecules. They are less well-suited (but still passable) for intermolecular interactions using the common two-body additive Coulomb interactions as used in Amber, Charmm, Gromacs. For these better choices would be amber99sb charges on proteins and peptides, and am1bccsym charges on the ligand.

### AM1 Charges

AM1 charges are a set of Mulliken-type charges derived from a semi-empirical quantum-mechanical calculation. For further discussion of this method, please see Dewar et. al. These should not be used for intermolecular interactions of force fields.

### AM1BCC Charges

AM1BCC charges start with Mulliken-type partial charges derived from the AM1 wave-function. In a second stage, bond-charge corrections (BCCs) are applied to the partial charges on each atom to generate new partial charges. Unlike "standard" AM1BCC charges, the default does not symmetrize charges (averaging those which are equivalent by bond topology, e.g. methyl hydrogens). This means that a multi-conformer molecule will not behave correctly with respect to conformational interconversion. For example, degenerate conformers will differ in energy, e.g. the 180-degree rotation of a carboxylate or amidinium. To get "standard" AM1BCC charges, specify the **am1bccsym** method.

It is important to avoid carrying out the AM1 calculation on a conformation with a strong short-range intramolecular polar interaction (e.g. a hydrogen bond or a salt bridge) because this significantly perturbs the AM1 starting charges used for AM1BCC, and the resulting AM1BCC charges would perform badly given a conformational change.

Due this issue, we incorporated into the recommended **am1bccsym** option the method proposed by Christopher Bayly and colleagues for assigning AM1BCC charges to a multiconformer molecule. It is based on the following procedure: the Coulomb electrostatic energy is calculated for every conformer using the absolute value of the MMFF94 partial charges (original negative charges are replaced with their absolute values). The standard AM1BCC calculation is then performed as above for the lowest electrostatic energy conformer determined in the previous step, and the AM1BCC charges obtained are assigned to all conformers.

The recommended **am1bccsym** option also carries out an AM1 geometry optimization lightly restrained to the starting coordinates is important to allow the relaxation of bond and bond angle degrees of freedom in AM1. Relatively small deviations of these from the AM1 optimum significantly affects the charges. Thus using the AM1 "single-point" option **am1bccspt** should be done with caution; bad input geometry will taint the charges. Thus, the partial charges produced from a single point calculation might, depending on the geometry, be similar to those from AM1 geometry optimized calculations but will not be as good quality.

OpenEye considers AM1BCC charges to be the best partial charge model currently available. For further discussion, please see the work of Christopher I. Bayly.

### Amber ff94, ff96, ff99, ff99sb, and ff99sbc0 Partial Charges

The partial charges used by the AmberFF94 force field are based on fitting quantum mechanical electrostatic potentials (esp). They were developed to address two key issues with earlier esp-fit charge sets: unrealistically high charges on charge centers and the variation of atomic charges with conformation. While the latter should have some basis in electronic structure, numerical instability in the charge fitting process was the source of both these pathologies. AmberFF94 charge sets use restrained esp-fitting (RESP) to control the numerical instabilities and simultaneous multi-conformer fitting to lead to conformation-independent charges that are restricted to individual residues. Particular

attention was given to ensure that backbone amides have consistent charges. The Amber force fields ff94, ff96, ff99, ff99sb, and ff99sbc0 all use the same set of RESP charges, they differ in other terms (mostly torsional).

## 3.9 *molcharge* Usage

*molcharge* uses the *-method* flag to determine what method to use to generate partial charges. Some methods can be applied to 2D input structures, others require 3D input structures; for the latter the run will fail if a 3D structure is not provided. Hydrogens will be sprouted on the molecules if they are not present already.

With default parameters, *molcharge* will sprout explicit hydrogens and assign MMFF94 partial charges.

### 3.9.1 Required Parameters

**-in**
: This is the input file. This file can contain molecules in a wide-variety of molecular formats. Some of the partial charging models, such as AM1 and AM1BCC require coordinates in order to calculate charges. While the input file is required, the *-in* flag is optional. If no *-in* flag is specified, the first unflagged parameter is used as the input file.

**-out**
: This is the output file and it is a required parameter. Since the object of *molcharge* is to generate partial charges, it will only write to formats that can specify a partial charge. These formats currently include only *.mol2*, *.mol2H* and *.oeb*. While the output file is required, the *-out* flag is optional. If no *-out* flag is specified, the second unflagged parameter is used as the output file.

#### Optional parameters

**-method** `{option}`
: Selects the charging method to be applied to the input molecule(s). Choose one of the following recommended options:

  **am1bccsym** applies AM1BCC charges as published, including partial semiempirical AM1 geometry optimization and averaging bond-topologically symmetric charges. It requires a 3D geometry associated with an input structure. These are good-quality charges for pharmaceutical organic molecules for intermolecular interactions.

  **mmff** applies MMFF94 charges. It will work with only a 2D geometry. This method is the best choice for use with the MMFF forcefield and these charges are of passable quality for intermolecular interactions. This is the default method.

  **amberff99sb** applies to proteins the RESP charge set used for Amber forcefields ff94, ff96, ff98, ff99, ff99sb, and ff99sbc0 (the same charge set is used in all these cases). This method only works for standard amino acids. These are very good-quality charges for proteins for intermolecular interactions.

  **gasteiger** applies gasteiger charges (a charge-equilibration method). **NOTE:** This method *should not be used for intermolecular interactions.* This method was intended for comparing relative reactivity of related organic chemical functional groups within different molecular contexts. It will work with only a 2D geometry.

  There are a number of other options for charge methods, but none are recommended. They are listed in the *molcharge* appendix.

**-param**
: The argument for this flag is the name of a file containing control parameters. The control parameter file acts to either replace or augment the command line interface. All parameters necessary for program execution may be provided in the control parameter file, although any command given explicitly on the command line will supersede options found in the parameter file. *molcharge* generates a new parameter file containing the full set

of execution parameters upon every execution. The name of the parameter file written by *molcharge* is created by combining the prefix base name with the *'.parm'* extension.

**-paramfile**
The filename for the output parameter file can be set with this flag and it overrides -prefix flag for the name of parameter file.

**-prefix**
The argument for this flag defines the prefix to be used for parameter and log files. The parameter and log files will be written to what follows this flag plus the extension *.param* or *.log*. [default = molcharge]

### 3.9.2 Example Commands

An example run of the *molcharge* program is given below.

```
prompt> molcharge -method mmff drugs.sdf drugs.mol2
```

This executes *molcharge* with the default parameters, to sprout explicit hydrogens and assign MMFF94 partial charges. The file *drugs.sdf* is opened in SD format for input, and the output is written to the file *drugs.mol2* in Sybyl .mol2 format.

```
prompt> molcharge -method amberff99sb 2iko_prot.pdb 2iko_prot.oeb
```

In this sample execution, a protein is charged using the Amber charge set and written to OEB format.

## 3.10 *molcharge* Appendix: Complete list of *-method* options

While there are many allowed options for the *-method* parameter in *molcharge*, only **am1bccsym**, **mmff** (Default), **amberff99sb**, and **gasteiger** are recommended (indicated in italics). The full list is given below:

**am1** applies the AM1 semiempirical method to derive the atomic charges for the molecule. A full geometry optimization is carried out. A 3D geometry for the input molecule is required.

**am1spt** applies the AM1 semiempirical method to derive the atomic charges for the molecule. Only a single-point calculation is carried out, so it is much faster but also very sensitive to the input geometry (bad charges from a bad geometry). A 3D geometry for the input molecule is required.

**am1bcc** is a synonym for **am1bccnosym** (described below).

---

**Note:** Note that this differs from the "standard" *AM1BCC* charging scheme as published, where symmetrization is done (same as **am1bccsym**).

---

**am1bccnosym** applies AM1BCC charges, but *without* averaging bond-topologically symmetric charges. The partial semiempirical AM1 geometry optimization is done. It requires a 3D geometry associated with an input structure. Non-symmetric charges can lead to problems with geometry optimizations and with multi-conformer molecules.

**am1bccsym** (*recommended*) applies AM1BCC charges as published, including partial semiempirical AM1 geometry optimization and averaging bond-topologically symmetric charges. It requires a 3D geometry associated with an input structure. These are good- quality charges for intermolecular interactions involving organic molecules.

**am1bccspt** is a synonym for **am1bccnosymspt** (described below).

**am1bccnosymspt** applies AM1BCC charges as published, but *without* averaging bond-topologically symmetric charges. Only a single-point AM1 calculation is carried out, so it is much faster but also

very sensitive to the input geometry (bad charges from a bad geometry). A 3D geometry for the input molecule is required.

**am1bccsymspt** applies AM1BCC charges as published, averaging bond-topologically symmetric charges. Only a single-point AM1 calculation is carried out, so it is much faster but also very sensitive to the input geometry (bad charges from a bad geometry). A 3D geometry for the input molecule is required.

**amberff99sb** (*recommended*) applies to proteins the RESP charge set used for Amber forcefields ff94, ff96, ff98, ff99, ff99sb, and ff99sbc0 (the same charge set is used in all these cases). This method only works for standard amino acids. These are very good-quality charges for intermolecular interactions involving proteins.

**am1bccelf10** charging is designed to average the AM1BCC charges from 10 conformers chosen from the 2% of the conformer population having the Electrostatically Least-interacting Functional (ELF) groups. 10 conformers from 2% means there must be at least 500 conformers to start with; ligands which have fewer rotatable bonds may not have this many. In such cases, this method is designed to take all the conformers in the 2% ELF population.

**amberff94**, **amberff99**, and **amberff99bsc0** are synonyms for **amberff99sb** The Amber forcefields use the same RESP charge set all cases

**gasteiger** (*recommended*) applies Gasteiger charges (a charge-equilibration method).

---

**Note:** This method *should not be used for intermolecular interactions.* This method was intended for comparing relative reactivity of related organic chemical functional groups within different molecular contexts. It will work with only a 2D geometry.

---

**mmff** (*recommended*) applies MMFF94 charges. It only requires a 2D geometry. This method ia the best choice for use with the MMFF forcefield and these charges are of passable quality for intermolecular interactions. This is the default method.

**initial** sets the charges to the MMFF94 initial fractional charges used for charged functional groups, smearing unit charges in the partial charge field onto resonance shared atoms.

**formal** copies the formal charge field of atoms into the partial charge field.

**none** sets the partial charges to zero.

CHAPTER

FOUR

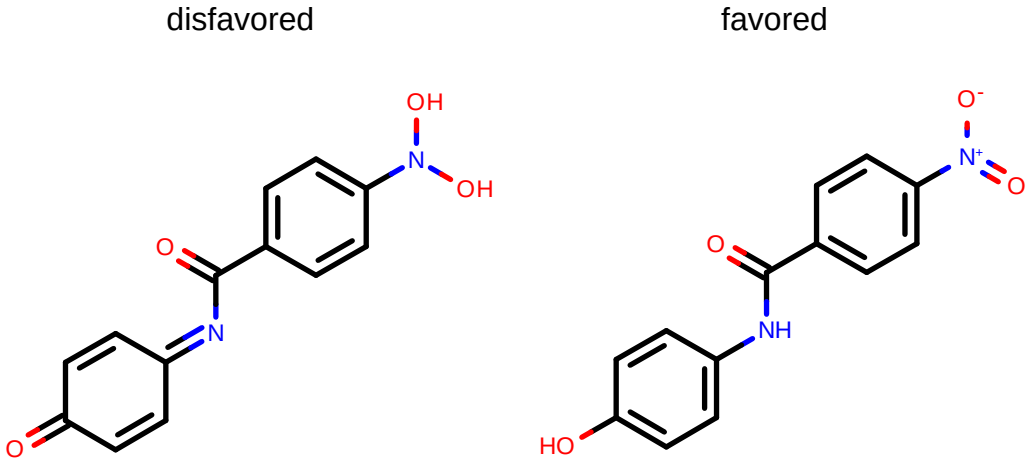
# RELEASE NOTES

## 4.1 Release History

### 4.1.1 QUACPAC 1.7.0 *(October 2016)*

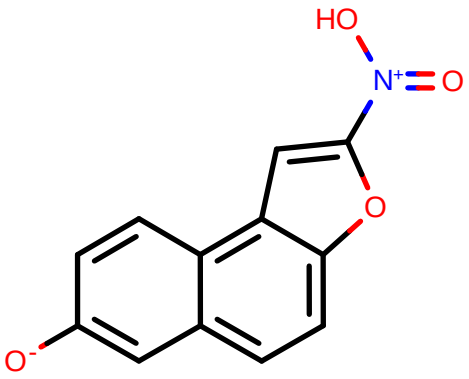
#### New features

- *-maxtime* flag has been added to tautomer enumeration for setting time limits on searching. Its default value is 60 seconds.
- *tautomers* has been dramatically improved to provide a low-energy, medicinally relevant, "reasonable" tautomeric form that is suitable for depiction for chemists. Significant improvements have been made to the reasonable tautomer algorithm that affect its aliphatic and non-aromatic resonance portions. The depictions below are a useful guide for recognizing how tautomers are favored as reasonable:
  - Conversion of carboxylates to diols and nitros to di-hydroxy amines is not favored.

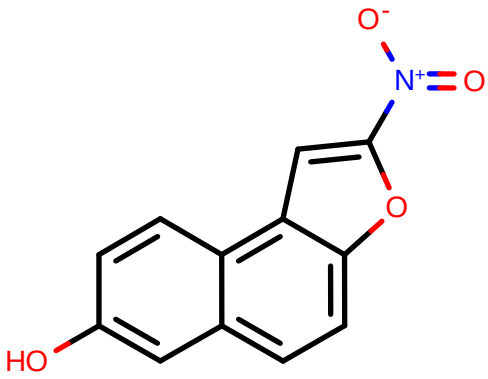



  - Generation of unnecessary, non-dative, formal charges is not favored.

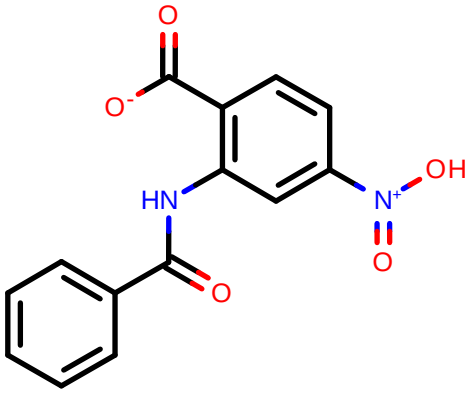
disfavored



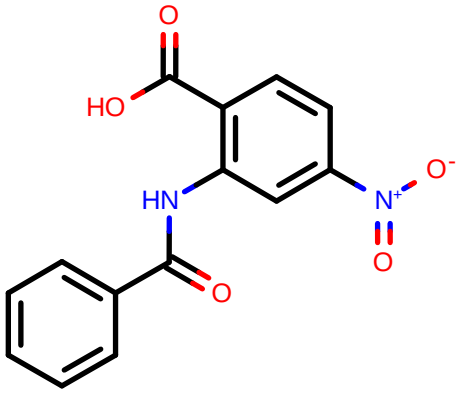

favored




disfavored

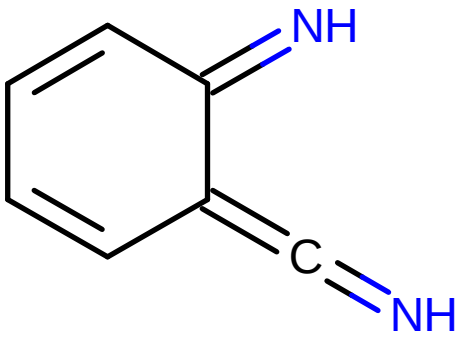



favored

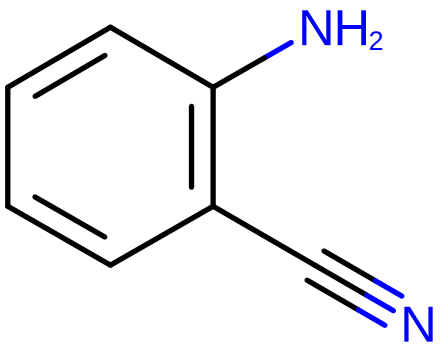



– Exocyclic bonds adjacent to aromatic rings are accounted for.

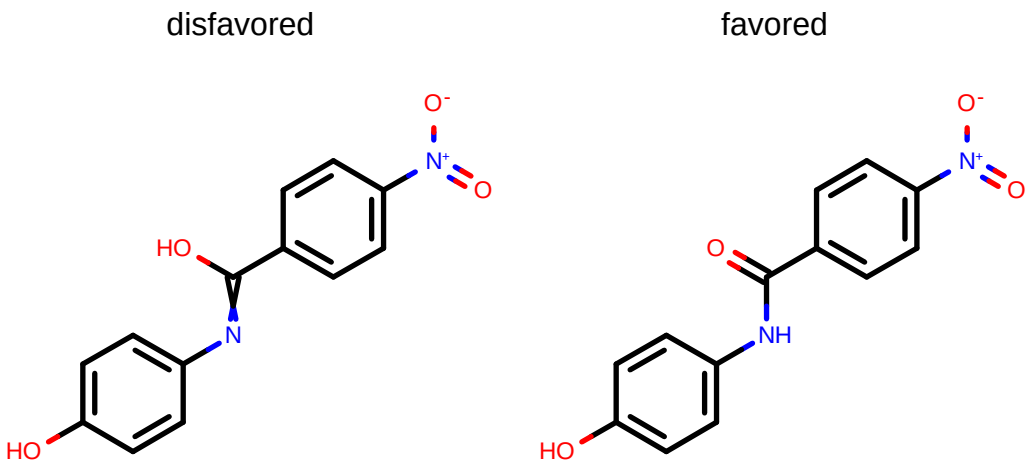
disfavored




favored



  - Priority is given to aliphatic double-bond positions.




- AM1BCC ELF10, a new method for applying partial charges to a molecule, has been added to *molcharge*. AM1BCC ELF10 allows up to 10 diverse conformers to be selected from those having the Electrostatically Least-interacting Functional groups (ELF). These conformers are then charged with the AM1BCCSym method and the charge sets are averaged to come up with a single charge set that is applied to all conformers. This yields good quality charge sets even for charged molecules.
- TIP3P water charges are now assigned when using Amber charge sets on molecules containing waters. The application of the other Amber charge sets to waters with explicit hydrogens will produce oxygen charges of -0.834 and hydrogen charges of +0.417.
- The default value of *-can* flag for *tautomers* has been changed to true.
- *tautomers* will output failed molecules to a fail file. The file will be named either *tautomers.fail* or, if the prefix is specified by *-prefix* flag, *prefix.fail*.
- A log file has been added for *molcharge*, *tautomers*, and *pkatyper*.
- *tautomers*, *molcharge*, *pkatyper*, and *fixpka* will now label unnamed input molecules as output_1, output_2, etc.

### Bug fixes

- *molcharge* now explicitly supports methods AM1BCCSym (a synonym for am1bcc) and AM1BCCSymspt (a synonym for am1bccspt).
- Descriptions for *molcharge* methods have been updated in the usage documentation. In addition, a description for *-formal* has been added.
- *pkatyper* no longer uses the deprecated OETyperMolFunction API.
- *tautomers* no longer uses the deprecated OETautomerMolFunction API.
- The wart symbol has been changed from "@" to "_" to help with parsing SMILES formatting.
- Unbounded stack allocations have been removed from *tautomers*.
- amberff94 will no longer identify a CYS residue as anionic when it is bonded by a sulfur to something other than another CYS residue.
- *pkatyper* no longer outputs an empty file if the output format is not supported.

#### Other changes

- opls method has been removed from *molcharge*.

### 4.1.2 QUACPAC 1.6.3 *(August 2013)*

#### New Features

- *molcharge* now uses the *-method* parameter for selecting a charge model. All individual flags for charge models have been removed.
- *tautomers* will now allow interconversion of [NH2+]= and [NH3+]-.
- *tautomers* now has a *-warts* option. Enabling this options will number the output molecules with an @ symbol.
- The default AM1BCC charge model in *molcharge* now lightly restrains the AM1 geometry optimization to the starting coordinates. This allows the important relaxation of bond and angle degrees of freedom while greatly reducing the potential to alter the molecule's conformation away from its starting coordinates.
- The AM1BCC charge models *-method am1bccsym* and *-method am1bccspt* in *molcharge* now symmetrize the partial charges over bond-topologically equivalent atoms, e.g. methyl hydrogens, in keeping with the original model. This is especially important with conformationally flexible molecules.
- *molcharge* now has a .fail file to output molecules that do not charge correctly.
- *fixpka* has been refined to reflect feedback from collaborators. In particular isoxazoles and oxadiazoles were added while pyrazoles and aryl sulfonamides were refined. Aryl sulfonamide refinement also incorporated changes based on newly obtained experimental data.

#### Bug Fixes

- *tautomers* will now properly set stereochemistry on planar carbon to *OEAtomStereo::Undefined*. Previously, planar carbon could inherit stereochemistry from an input molecule with stereochemistry set on a tetrahedral carbon.
- A bug has been fixed in *tautomers* where 3D hydrogens where being suppressed.
- *fixpka* now correctly addresses beta di-amino groups so that if all other considerations are equivalent, secondary amines are treated as most basic, followed by primary amines, with tertiary amine being treated as least basic.
- A few bugs have been fixed in *fixpka* involving quaternary nitrogen and protonated aromatic nitrogens.
- *tautomers* has been fixed so that 3D molecules have 3D hydrogens instead of implicit hydrogens.
- *molcharge* no longer inadvertently converts reduced cysteine residues (CYS) to the oxidized form (CYX) when assigning Amber partial charges to protein residues.

#### Other changes

- The *-all* flag has been removed from *tautomers*. Equivalent results can been achieved with the following options: *-level 7 -ch3 true*. Level 7 will likely yield a plethora of very unreasonable tautomers and is not recommended.
- This will be the last release to support SuSe 10.

### 4.1.3 QUACPAC 1.5.0

#### New Features

- *fixpka* has been added. This application may be used to set a molecule to an energetically favorable ionization state for pH=7.4. This is the same pH model that was available in the Filter application. Additionally, the perception of acceptable valence states has been improved to include phosphorus as well as aromatic oxygen and sulfur with +1 formal charge.
- In *tautomers*: Major improvement for carbon hybridization. Now carbon atoms with bonds to one, two, or three heavy atoms are able to change hybridization state. However, only carbons close to an appropriate heteroatom are allowed to change hybridization state. Additionally, unreasonable charge states of carbon have been removed. Now the enabling the *-ch3* option will only generate tautomers appropriate for the given level.
- In *tautomers*: Improvement of the *-reasonable* option, particularly involving exocyclic heteroatoms.
- In *tautomers*: Added stereo preservation flag *-savestereo* with a default value of *false*. Stereo chemistry can be lost during creation and removal of double bonds, but if the user desires a certain stereo setting to be preserved this flag will prevent the associated atoms and bonds from taking part in tautomerization.
- *pkatyper* now implicitly uses the *-reasonable true* setting for its internal call to *tautomers*. This provides a better reference tautomer when enumerating pKa states.
- In *pkatyper*: Improvements to pka states.
- In *molcharge*: Aromaticity settings on molecules are now unchanged on returned molecules. Aromaticity may be temporarily changed inside the function while charges are being calculated, but the molecule will have the same aromaticity after the function as before.
- In *molcharge*: AmberFF94 charges can now be applied to standard protein residues.
- In *molcharge*: A new *-altlocs* flag will retain all alternative locations in a disordered .pdb structure when charging. This option allows dictionary-based charges such as AmberFF94 to be applied to all atoms in a disordered molecule (but should not be used with conformation-dependent charges such as AM1BCC).

#### Bug Fixes

- In *tautomers*, atom-ordering could in rare instances be inconsistent for output molecules. This led to cases where even though all associated tautomers led to the same unique tautomer, the unique tautomer could have different SMILES representations depending on the input structure. This has been fixed.

### 4.1.4 QUACPAC 1.3.1

#### New features

- The distribution and installation of QUACPAC has been modified. The Windows distribution is now a standard EXE installer, and the OS X distribution is a dmg containing a standard pkg installer. The executables are now scripts that chose the correct version of the program at runtime. Please see the Application Installation section for details.

#### Bug fixes

- The previous version had a license failure when AM1 charges were calculated. This bug has been fixed.

### 4.1.5 QUACPAC 1.3.0

This release represents the first major reworking of the QUACPAC toolkits and applications in several years. Any users of prior versions of QUACPAC will quickly notice several significant changes. First, this version of QUACPAC does not contain a new release of *protein_pka* program. We are in the midst of a major rewrite of the *protein_pka* program. We hope that this work will bring major improvements in the usability, science and analysis of the *protein_pka* program, yet we do not want to delay the release of the entire QUACPAC package waiting for the *protein_pka* program. The current QUACPAC release does not contain *protein_pka*, but it will be included in a future release when the rewrite is complete. In the interim, we hope you find the bug fixes and new features included in this release useful.

The preps and qpd programs will no longer be supported future versions of QUACPAC.

#### New features

- A reasonable looking tautomer may be calculated by using the *-reasonable* flag.
- A multiconformer method has been added for calculating AM1BCC charges

#### Bug fixes

- SMILES map indexes and names are no longer lost when outputting molecules.

### 4.1.6 QUACPAC 1.1.0

- Special Note: Version 1.4 of *molcharge* and version 1.2b of the library have been released and inserted into version 1.1 of QUACPAC. Both of these have removed support for the VC2003 partial charging method.
- Version 1.1 is the first full stable release of QUACPAC. QUACPAC remains a heterogeneous release including five primary applications and a programming library with a C++ api.
- *tautomers* is in version 2.0. It provides enumeration of energetically reasonable tautomers of input molecules.
- *pkatyper* is in version 1.1. *pkatyper* provides enumeration of protonation states (pKa ~2-~14).
- *molcharge* is in version 1.3. *molcharge* provides MMFF, AM1, AM1BCC, VC2003 and other partial charges on small molecules.
- *protein_pka* is in version 1.3. *protein_pka* carries out PB calculations to assess the shifts in protein residue pKa's.
- This release includes the oeproton library. The oeproton library exposes the features of *pkatyper*, *tautomers* and *molcharge* in three high level C++ api points. The version of the library in this release is 1.1b. However, the beta moniker signifies only that at this time, OpenEye reserves the right to modify this api. We believe the quality of the code in this library is very solid and the beta designation does not reflect its quality.

## 4.2 Citation

**Note:** To cite QUACPAC please use the following:

QUACPAC 1.7.0.2: OpenEye Scientific Software, Santa Fe, NM. http://www.eyesopen.com.

[Alexov-1999] E. Alexov and M.R. Gunner,
**Incorporating Protein Conformational Flexibility into pH-titration Calculations: Results on T4 Lysozyme**,
*Biophysical Journal*, Vol. 74, pp. 2075-2093, **1999**.

[Baker-1934] John W. Baker, *Tautomerism*, D. Van Nostrand Company Inc.
Publishers, **1934**.

[Dewar-1985] M.J.S Dewar, E.G. Zoebisch, E.F. Healy and J.J.P. Stewart,
**AM1: A New General Purpose Quantum Mechanical Model**,
*Journal of the American Chemical Society*, Vol. 107, pp. 3902-3909, **1985**.

[Elguero-1976] Jose Elguero, Claude Marzin, Alan R. Katritzky and Paolo Linda,
**The Tautomerism of Heterocycles**,
Supplement 1, *Advances in Heterocyclic Chemistry*, Academic Press, **1976**.

[Ertl-1997] Peter Ertl,
**Simple Quantum Chemical Parameters as an Alternative to the Hammett Sigma Constants in QSAR Studies**,
*Quantitative Structure-Activity Relationships (QSAR)*, Vol. 16, pp. 377-382, **1997**.

[Gasteiger-1978] J. Gasteiger and M. Marsili,
**A New Model for Calculating Atomic Charges in Molecules**,
*Tetrahedron Letters*, pp. 3181-3184, **1978**.

[Gasteiger-1980] J. Gasteiger and M. Marsili,
**Iterative Partial Equalization of Orbital Electronegativity - A Rapid Access to Atomic Charges**,
*Tetrahedron*, Vol. 36, pp. 3219-3228, **1980**.

[Jakalian-2000] Araz Jakalian, Bruce L. Bush, David B. Jack and Christopher I. Bayly,
**Fast, Efficient Generation of High-Quality Atomic Charges. AM1-BCC Model: I: Method**,
*Journal of Computational Chemistry*,
Vol. 21, pp. 132-146, **2000**.

[Jakalian-2002] Araz Jakalian, David B. Jack and Christopher I. Bayly,
**Fast, Efficient Generation of High-Quality Atomic Charges. AM1-BCC Model: II: Parameterization and Validation**,
*Journal of Computational Chemistry*, Vol. 23, pp. 1623-1641, **2002**.

[Katritzky-2000] Alan R. Katritzky and A.F. Pozharskii,
**Handbook of Heterocyclic Chemistry**,
*Academic Press*, 2nd Edition, **2000**.

[McGuire-2000] A. Ting, R. McGuire, A.P. Johnson and S. Green,
**Expert System Assisted Pharmacophore Identification**,
*Journal of Chemical Information and Computer Science (JCICS)*, Vol. 40, No. 2, pp. 347-353,
**2000**.

[Sayle-1999] Roger Sayle and Jack Delany,
**Canonicalization and Enumeration of Tautomers**, in
*Innovative Computational Applications*,
Institute for International Research, Sir Francis Drake Hotel,
San Francisco, 25-27 October **1999**.

[Talgren-1996] T.A. Halgren,
**Merck Molecular Force Field: II. MMFF94 van der Waals and Electrostatic Parameters for Intermolecular Interactions**,
*Journal of Computational Chemistry*, Vol. 17, No. 5, pp. 520–552, **1996**.

[Trepalin-2003] Sergey V. Trepalin, Andrey V. Skorenko, Konstantin V. Balakin,
Anatoly F. Nasonov, Stanley A. Lang, Andrey A. Ivashchenko and
Nikolay P. Savchuk,
**Advanced Exact Structure Searching in Large Databases of Chemical Compounds**,
*Journal of Chemical Information and Computer Science (JCICS)*, Vol. 43, No. 3, pp. 852–860, **2003**.

[Yang-1993] Yang, A.-S., M. R. Gunner, R. Sampogna, K. Sharp and B. Honig,
**On the Calculation of pKa's in Proteins**, *Proteins*, Vol. 15,
pp. 252-265, **1993**.

## Symbols



## A



## C



## E



## O



## P