



OpenEye
Scientific Software

ToolkitRelease – Python

Release 1.7.2

OpenEye Scientific Software, Inc.

November 02, 2009

CONTENTS

1	OEToolkits 1.7.2	1
2	OEChem 1.7.2	3
2.1	New features	3
2.2	Major bug fixes	4
2.3	Minor bug fixes	5
3	OESystem 1.7.2	7
3.1	New features	7
3.2	Major bug fixes	7
3.3	Minor bug fixes	7
4	OEPlatform 1.7.2	9
4.1	New features	9
4.2	Minor bug fixes	9
5	OEBio 1.7.2	11
5.1	New features	11
5.2	Major bug fixes	11
6	OEGrid 1.3.3	13
6.1	New features	13
6.2	Major bug fixes	13
7	OEGraphSim 1.0.0	15
8	Lexichem 1.2.0	17
9	OEMolProp 2.1.0	19
10	Ogham 1.7.1	21
11	OmegaTK 2.4.0	23
11.1	New features	23
11.2	Bug fixes	23
12	QuacPacTK 1.4.1	25
12.1	Bug Fixes	25
13	ShapeTK 1.7.2	27
13.1	New features	27

13.2	Bug fixes	27
14	SpicoliTK 1.1.0	29
14.1	New Features	29
14.2	Bug fixes	29
15	SzybkiTK 1.4.0	31
15.1	New features	31
15.2	Bug fixes	31
16	ZapTK 2.1.1	33
16.1	Bug fixes	33
16.2	Minor bug fixes	33

OETOOLKITS 1.7.2

This is a new release of the OpenEye Toolkits with versions of the following libraries:

OEChem TK 1.7.2
GraphSim TK 1.0.0
Grid TK 1.3.3
Lexichem TK 2.0.0
MolProp TK 2.1.0
Ogham TK 1.7.1
Omega TK 2.4.0
Quacpac TK 1.4.1
Shape TK 1.7.2
Spicoli TK 1.1.0
Szybki TK 1.4.0
Zap TK 2.1.1

- There are two major additions to the OpenEye toolkit suite with this release:

GraphSim TK

A toolkit for doing 2D molecule similarity. The methods currently implemented are:

- Path based fingerprints
- MACCS key fingerprints
- LINGOS

MolProp TK

A toolkit derived from OpenEye's FILTER technology. This toolkit is useful for calculating an extensive list 2D molecular properties and building custom filters around these properties.

Note: The version number of *MolProp TK* will track with the FILTER application to indicate similarities.

- This release of OEChem introduces many bug fixes and new features provided they didn't break source-level compatibility with 1.7.0.
- This release completes the transition from the old LaTeX based documentation system to the new reStructured-Text based system. This includes better support for Python and Java in the manuals as separate manuals for every library.

The following release notes are only for the python toolkit:

- The `OE_ARCH` environment variable has been introduced to force the python libraries to load shared libraries for a specific architecture. This can be useful for getting OpenEye libraries to work on non-officially-supported platforms.
- Quacpac: Added support for writing a custom `OETautomerMolFunction` or `OETyperMolFunction` in Python by deriving from the base class.

OEChem 1.7.2

- `OE_PDBIFlag`, `OE_PDBOFlag`, `OE_MDLOFlag`, `OE_MOPACFlag` constants namespaces are deprecated.

2.1 New features

- `OEReadMDLQueryFile` now sets the title of the query molecule.
- *OEChem* now recognizes the MDL “wavy” bond. An `OEBondBase` can be queried for its stereo type by querying its generic data for the `OEProperty_BondStereo` tag. The returned unsigned int will be from the `OE_BondStereo` namespace.
- *OEChem* now supports MDL query file R group definitions. These are lines starting with `M RGP`. The R group is represented as a `OEQAtomBase` with a `OEAtomBase.GetMapIdx` the same as the R group number in the file.
- *OEChem* will now correct amidine and guanidinium functional groups where the bonds between C and N are marked as aromatic in MOL2 files.
- Significant performance improvement to reading single-conformer molecules from OEB files. Other minor performance improvements were made to OEB as well (including multi-conformer molecules). This speeds up `OEDBMol` compression and uncompression as well.
- Removed `OECopyHistory` function. `OEReadHeader` is now smart enough to read the header into the `OEHeader` history if it is already populated with another header’s information.
- `oemolthreads` now support reading and writing `OEHeader` objects.
- Added the `oemolthreadbase.PeekMol` method.

The following release notes are only for the python toolkit:

- Support for the copy module interface (`__copy__` and `__deepcopy__`) for objects that inherit from `OEBase`. This makes *OEChem* molecules behave a little more pythonic allowing the user to care a little less that they are C++ based objects.
- Added support for `OEBaseData.GetData`. This improves the ability to use `OEBaseData.AddData` and `OEBaseData.GetDataIter` from python.
- Added partial support for `OEBase.AddData` in python. Only primitive data types will work. More complex data types (Molecules, Surfaces, etc) will fall back to `OEBase.SetData`.
- `OE_SmartsLexReplace` is now properly wrapped for python.

- Added `OEAtomArray` and `OEBondArray` classes (similar to `OEFloatArray`) for dealing with arrays of atom and bond pointers from python.
- Fixed an inconsistency of using `OEIter.Target`. Prior to 1.7.1 if `Target` was used inside a python iteration the `Target` would point to the next item in the iterator, not the current one. Therefore, the following assertion always holds:

```
atomiter = mol.GetAtoms()
for atom in atomiter:
    assert atom == atomiter.Target()
```

This fix was implemented to support lazy iterators as returned by the new alternate location utilities: `OEAltLocationFactory.GetGroups` and `OEAltGroup.GetLocations`.

Note:

Since the fix was implemented as a reimplement of the `OEIter.__iter__` method to make it return a python generator instead of the iterator object itself there is now a slight difference in the above code snippet and the alternative way of using the `OEIter.next` python iteration protocol as shown in the following code:

```
atomiter = mol.GetAtoms()
try:
    while True:
        atom = atomiter.next()
        assert atom != atomiter.Target()
except StopIteration:
    pass
```

This previous code snippet will behave the same in 1.7.0 and 1.7.1. However, note that now `OEIter.Target` returns the next item in the iterator. This is fundamentally how `OEIter.next` is required to work.

We could have not provided `OEIter.next` at all since the only requirement for python iteration is the `__iter__` method. However, we choose to leave a `OEIter.next` implementation in order to not break code that was written like the following:

```
atomiter = mol.GetAtoms()
atom = atomiter.next() # grab the first atom
```

2.2 Major bug fixes

- Fixed a `OEReadMDLQueryFile` segmentation fault when perceiving aromaticity with spirane rings containing generic atoms.
- Fixed a bug where atom parity was ignored when reading a 3D MOL file.
- Fixed a memory leak when initializing an `OEMCSSearch` with its constructor. The workaround in 1.7.0 is to use the `OEMCSSearch.Init` method instead.
- Fixed a memory leak with `OEMCSSearch` when `OEMCSSearch.SetMaxMatches` was set to a number lower than the total number of matches found.
- Fixed memory leak when creating and destroying multiple `OELingoSim` objects.
- Some `OEOFlavor_PDB` flavor combinations would generate corrupt data.
- `OESmilesAtomCount` now includes atoms of the form `[#6]` in its count.

- Fixed a bug where map indices on explicit hydrogens would be ignored when generating SMILES. For example, `[CH]` would get generated even if the hydrogen had a map index specified. Now `[H:1][C]` will be generated.

The following release note is only for the python toolkit:

- The feature added in 1.7.0 of retrieving a python dictionary of the contents of an `OEBase` with the `OEBase.GetData` method was leaking memory. This has been fixed in this release.

2.3 Minor bug fixes

- Energy set on an `OEGraphMol` using `OEMolBase.SetEnergy` would not get written out to OEB files. Energy data will now round-trip through the OEB format.
- Automatic conformer combining through the `oemolistream.SetConfTest` interface will now combine separate single-conformer molecules in an OEB into a multi-conformer molecule similar to the other file formats *OEChem* supports for that behavior (MOL2, SDF, etc).
- `OEPerceiveBondOrders` is more stable to pre-existing bond orders on the molecule.
- Rare non-deterministic bug fixed for sulfur monoxide bug in `OEPerceiveBondOrders`.
- *OEChem* now recognizes the DU residue name and the mythical DT residue which is synonym for “T”.
- The new overload of `OEIsReadable` to take a string would return `true` if there was no periods (‘.’) in the string and the string began with a readable file extension, for example, “MDL-FOO”.
- Fixed a bug when `OELibraryGen` failed to generated products with correct Kekulé-form.

OESYSTEM 1.7.2

3.1 New features

- Added the `OEGetBitCounts` function to efficiently get certain bit counts when comparing two `OEBitVector` objects. This function is useful for implementing custom fingerprint similarity measures.
- Added the `OEBitVector.SetData` method to allow a rapid method of initializing an `OEBitVector` with binary data.
- Added `OEErrorHandler.Verbose` method to send `OEErrorLevel_Verbose` level messages.
- Added `OEHeader.operator bool` to test whether the `OEHeader` has any data in it.
- Added `OEBuffer.Peek` and `OEProtectedBuffer.Peek` methods.
- Optimized writing generic data attached to an `OEBase`. This optimized `OEBase::Compress`.
- Minor optimization to a common *OEChem* iterator.

3.2 Major bug fixes

- Fixed a memory leak with `OEConstIter`.

3.3 Minor bug fixes

- Renamed the `OEMakeFP` function to `OEParseHex` to be more explicit about what it does.
- Valgrind would throw warnings about a change made in 1.7.0 for thread safety issues. Though this was not a true memory leak the code has been slightly changed to silence this warning.
- Fixed an exponential growth bug when adding history to an `OEHeader`.
- Assignment operators for `OEMultiGrid` now properly copy the titles of the subgrids.

OEPLATFORM 1.7.2

4.1 New features

- Added the `OESetLicenseFile` function to allow dynamically setting which OpenEye license file to use.
- Added the `OEGetCurrentWorkingDirectory` function to return the current working directory for the process.
- Optimized `oeigzstream.seek` to only rewind the stream if seeking backwards. Therefore, the optimal way to seek in a gzipped file is in increasing `oefpos_t` order.
- Optimized `oeisstream.getbuffer`.
- Optimized `oeosstream.clear`.

4.2 Minor bug fixes

- Revised the `oestream.length` and `oestream.size` documentation to be more accurate.
- `oefpos_t` nows works for files greater than 4GB on 64-bit windows.
- Fixed an infinite loop that would occur from the following code:

```
oeosstream sfs;  
sfs.open("foobar");  
sfs.write("blah");
```

`oeosstream.open` will return `false`. Ignoring the `false` return value would cause `oeosstream.write` to loop indefinitely.

OEBIO 1.7.2

5.1 New features

- Introduced new *OEBio* classes to manage alternate locations in protein data bank structures along with a new PDB input flavor `OEIFlavor_PDB_ALTLOC` to retain alternate location atoms on input.

5.2 Major bug fixes

- The crystal symmetry routines were broken in the 1.7.0 release by a corrupted matrix.
- When reading symmetry from external sources (*i.e.* PDB files or maps) a warning is thrown when reading out of date space groups rather than fail to read symmetry.
- Setting space groups will fail unless the most current space group constraints are used.
- Added the following space group aliases for older style space groups.

```
I 1 2 1 -> C 1 2 1  
P 1-    -> P -1
```


OEGRID 1.3.3

6.1 New features

- `OEMakeGridFromCenterAndExtents` added for a grid of an arbitrary template type.

6.2 Major bug fixes

- Fixed a rare crash in `OEMakeRegularGrid` that was caused by floating point round-off error.

OEGRAPHSIM 1.0.0

- Provides the following features:
 - Generate three basic fingerprint types (MACCS key, LINGO and path-based)
 - Parameterize the path-based fingerprint generation
 - * allowing various atom/bond typing
 - * specifying the minimum and maximum length of the enumerated paths
 - * setting the size of fingerprint
 - Store and retrieve fingerprints as generic data
 - Save/import fingerprints to/from OEB binary file format
- Implementation of six common similarity indices (Cosine, Dice, Euclidean, Manhattan, Tanimoto, Tversky)
- Implementation of fingerprint database that provides the following methods for rapid fingerprint-based similarity search:
 - apply user-defined similarity measures
 - set similarity cutoff value
 - access similarity in sorted order

LEXICHEM 1.2.0

- This release includes the ability to parse stereo on input names. Previously it was read and ignored.
- Fixed a bug where, in rare cases, the output name depended on input atom ordering.
- Fixed a crash in determining CIP stereo for very large, pathological molecules.

OEMOLPROP 2.1.0

This is the first release of the OEMolProp toolkit. Its version number corresponds to version number of the Filter application.

OGHAM 1.7.1

- Adding more than 400 common macro-cycle ring systems to the ring template dictionary

OMEGATK 2.4.0

11.1 New features

- The `MaxConfGen` and `MaxPoolSize` variables are now set internally and adjusted automatically subject to the user defined variable of `MaxConfs`. Additionally, hard limits have been removed from these variables and they are now only limited by available memory. The following methods are deprecated: `OEOmega.SetMaxConfGen`, `OEOmega.GetMaxConfGen`, `OEOmega.SetMaxPoolSize`, `OEOmega.GetMaxPoolSize`
- The method `OEOmega.SetMaxConfs` has been changed from *void* to *bool*. The method will return *true* if the parameter has been set properly and will return *false* if the number is too high and would cause the memory required for duplicate removal to exceed the maximum theoretical memory for the platform.
- The method `OEOmega.SetFixSmarts` has been created to allow a smarts pattern to be used to fix a portion of the molecule. This requires 3D coordinates to be available from either a 3D input file with `OEOmega.SetFromCT` set to *false* or from a molecule set using `OEOmega.SetFixMol`.
- Fragment generation for `OEOmega` is now the same as the `makefraglib` application. Previous versions used a faster but more approximate version of fragment generation.
- The `OEOmega` default has been changed to no longer use “close enough” atom typing. If a proper MMFF atom type cannot be found for an atom then the molecule will fail. Previous versions would use a different atom type of the same element if it was available. The previous behavior of atom type substitution can be enabled by calling `OEOmega.UseStrictAtomTypes` with the argument *false*.
- The error level for most messages coming from the OmegaTK library has been lowered to *Verbose*. The library can be run in a much quieter manner if the error level is set to *Info* by the user.

11.2 Bug fixes

- Crashes have been fixed that could occur when the `FixFile` could not find a valid match or did not have enough atoms to match against in a ring system.
- Program no longer crash if there is not enough memory for duplicate removal. If there is not enough memory available for allocation then duplicate removal will be skipped for the molecule and `OEOmega.operator()` will return *true*.

QUACPACTK 1.4.1

12.1 Bug Fixes

- `OESetNeutralpHModel` has been added to the Java and Python wrappers.
- Bug has been fixed in `OEAssignPartialCharges` where it sometimes did not return false after a failure to assign charges. Now this function should always return false if charges are not set.
- Fixed tetrionic acid rule in `OESetNeutralpHModel` to allow matching of aromatic bonds.

SHAPETK 1.7.2

13.1 New features

- `OEAddColorAtoms` now has an optional third argument to cause precalculation and caching of the self color. When re-using molecules in multiple comparisons (for example NxN studies), use `OEAddColorAtoms` once and then `OEBestOverlay` and `OECOLOROverlap` will use the pre-stored color atoms and self color to speed up the calculations.
- Added a set of functions to manipulate cached self color.
 - `OESetCachedSelfColor`
 - `OEHasCachedSelfColor`
 - `OEGetCachedSelfColor`
 - `OEClearCachedSelfColor`
- The pair of functions (`OECOLORAtomsToString` and `OESTringToColorAtoms`) added in v1.7.1 have been renamed to more appropriate names. The new functions are `OECompressColorAtoms` and `OEUncompressColorAtoms`.

13.2 Bug fixes

- Fixed a bug where dummy atoms could cause an erroneous result for `OECALCVolume`.

SPICOLITK 1.1.0

14.1 New Features

- Added the `OEMakeBitGridFromSurface` function. This is a lot faster than using `OEMakeGridFromSurface` to generate a lookup grid for whether a point is inside or outside a surface.

14.2 Bug fixes

- The previous release (1.0.2) introduced the feature that didn't require the user to call `OEInitSurfaceHandlers`. This didn't work sometimes when statically linking an executable. This was because the library initialization code that needed to get run was getting discarded by the linker because none of the functions in that translation unit were being called. The library initialization code has been migrated to the same translation unit as the `OESurface` since these functions always have to be called to attach a molecule as generic data.

Note: This only affected C++ users.

SZYBKITK 1.4.0

15.1 New features

- Starting from the current release, certain classes of divalent selenium compounds can be handled by Szybki. Currently only some compounds in which selenium is bonded to the hydrogen or carbon atoms can be processed. Those compounds include selenols (RSeH) and selenides (RSeR1) in which Se is bonded to alkyl or aromatic carbon atoms, with the following exceptions: - Se makes a bond with alkyl C which belongs to the 3- or 4-membered ring - Se makes a bond with aromatic C which belongs to 5-membered ring In the restriction above MMFF94 aromaticity model is adopted, so for example a pyridine derivative, pyridine-3-selenol can be processed, but a pyran derivative 2H-pyran-3-selenol cannot.
- Previous Szybki releases attempted to assign the “closest” MMFF94 atom type to those atoms for which a unique MMFF94 type could not be found. Starting from the 1.3.5 release this feature is removed, so consequently the input molecules containing such atoms will not be processed.
- Ligand RMSD which might be extracted from OESzybkiResults object, can optionally refer to heavy atoms only. A new public function: void SetLigandRMSDHeavy(bool) in OESzybki class has been added for that purpose.

15.2 Bug fixes

- A bug causing memory corruption when the function SetSideChainsFlexibility() was called after SetProten() was fixed.
- Memory leak in ligand torsion optimization inside the partially flexible protein receptor was removed.

ZAPTK 2.1.1

16.1 Bug fixes

- `OEZap.GetMolecule` and `OEZap.GetFocusTarget` now return a pointer instead of a reference. This is a breaking change and any code that calls these methods will have to be updated. These methods will return a null pointer if an acceptable molecule has not been set for them.
- A warning has been added for molecules passed into ZapTK that do not return 3 when `GetDimension` is called. The dimension must be 3 for all molecules passed into ZapTK. The dimension is set automatically when using `OEReadMolecule` but a warning will be thrown and the molecule will not be accepted by ZapTK if the molecule has been made from scratch in the toolkits and `SetDimension(3)` has not been called on it.

16.2 Minor bug fixes

- Fixed a memory bug related to extremely large files on Windows.