



**OpenEye**  
Scientific Software

**EON**

*Release 2.1.0*

**OpenEye Scientific Software, Inc.**

June 03, 2011



# CONTENTS

<b>1</b>	<b>Front Matter</b>	<b>1</b>
<b>2</b>	<b>Installation and Platform Notes</b>	<b>3</b>
2.1	Licenses . . . . .	3
2.2	General Installation . . . . .	3
2.3	PVM . . . . .	5
2.4	Open MPI . . . . .	6
<b>3</b>	<b>Usage</b>	<b>9</b>
3.1	Introduction . . . . .	9
3.2	Command Line Interface . . . . .	9
3.3	Report File . . . . .	14
<b>4</b>	<b>Theory</b>	<b>15</b>
<b>5</b>	<b>Release Notes</b>	<b>17</b>
5.1	EON v2.1.0 . . . . .	17
5.2	EON v2.0.0 . . . . .	17
<b>6</b>	<b>List of Selected EON Publications</b>	<b>19</b>
	<b>Bibliography</b>	<b>21</b>
	<b>Index</b>	<b>23</b>



# FRONT MATTER

Copyright 1997-2011 OpenEye Scientific Software, Santa Fe, New Mexico. All rights reserved.

All rights reserved. This material contains proprietary information of OpenEye Scientific Software. Use of copyright notice is precautionary only and does not imply publication or disclosure.

The information supplied in this document is believed to be true but no liability is assumed for its use or the infringement of the rights of others resulting from its use. Information in this document is subject to change without notice and does not represent a commitment on the part of OpenEye Scientific Software.

This package is sold/licensed/distributed subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out or otherwise circulated without OpenEye Scientific Software's prior consent, in any form of packaging or cover other than that in which it was produced. No part of this manual or accompanying documentation, may be reproduced, stored in a retrieval system on optical or magnetic disk, tape, CD, DVD or other medium, or transmitted in any form or by any means, electronic, mechanical, photocopying recording or otherwise for any purpose other than for the purchaser's personal use without a legal agreement or other written permission granted by OpenEye.

This product should not be used in the planning, construction, maintenance, operation or use of any nuclear facility nor the flight, navigation or communication of aircraft or ground support equipment. OpenEye Scientific Software, shall not be liable, in whole or in part, for any claims arising from such use, including death, bankruptcy or outbreak of war.

Windows is a registered trademark of Microsoft Corporation. Apple, OS X, and Macintosh are registered trademarks of Apple Computer, Inc. AIX and IBM are registered trademarks of International Business Machines Corporation. UNIX is a registered trademark of the Open Group. RedHat is a registered trademark of RedHat, Inc. Linux is a registered trademark of Linus Torvalds. SPARC is a registered trademark of SPARC International Inc.

SYBYL is a registered trademark of TRIPOS, Inc. MDL is a registered trademark and ISIS is a trademark of Accelrys, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrodinger, Inc.

Python is a trademark of the Python Software Foundation. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. and other countries.

Other products and software packages referenced in this document are trademarks and registered trademarks of their respective vendors or manufacturers.



---

# INSTALLATION AND PLATFORM NOTES

## 2.1 Licenses

To run EON you will need to obtain a license file for EON from OpenEye Scientific Software ([business@eyesopen.com](mailto:business@eyesopen.com)). The license file should be pointed to by the environment variable **OE\_LICENSE**.

If you intend on running multi-processor EON via PVM or Open MPI, only the master machine needs access to the license file.

On Windows, the environment variables can be set under the system Control Panel.

## 2.2 General Installation

By default, all OpenEye applications are installed into a single distribution directory tree on the specified machine. The default location for this tree is platform specific and will be detailed below.

The root of the tree (i.e. the `openeye` directory) contains the following subdirectories:

- admin** This directory is intended to contain any administrative scripts and tools associated with the installed applications. Currently, this directory is simply a placeholder on all platforms except for Microsoft Windows, where it contains the uninstaller executables.
- 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-RHEL5-x64:redhat-RHEL4-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:

- |                     |  |
|---------------------|--|
| <b>-path</b>        | 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.  |
| <b>-print_arch</b>  | 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. |
| <b>-use_version</b> | Specifying this argument followed by a specific version number allows the user to control which released version of the executable to run.   |

## 2.2.1 Linux/Unix

Linux/Unix distributions are provided as a gzipped tarball of the distribution tree described above. Installation is performed by untarring the file in the desired location. Multiple distributions can be installed in the same location without any challenge.

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`.

## 2.2.2 Windows

Windows distributions are provided as a standard EXE installer. By default, all OpenEye applications will install into the `C:\OpenEye` directory.

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, as well as to a Windows command shell which has

the appropriate **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.

For GUI 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 standard *pkg* installer delivered as a *dmg* disk image. By default, all OpenEye applications will install into the `/Applications/OpenEye` directory.

To ensure that the installed applications can be called from the command line in the *Terminal*, be sure to add `/Applications/OpenEye/bin` to the **PATH** environment variable.

For GUI applications, an application bundle which can be clicked on to start, will be present in the `/Applications/OpenEye` directory. This bundle cannot be moved independent of the `OpenEye` directory. For instance, the entire `OpenEye` directory can be moved as one piece, but moving the application bundle or the contents of any of the subdirectories in the `OpenEye` directory may cause the application to not start. However, the bundle can still be dragged into the Dock and run from there without any problem.

## 2.3 PVM

PVM or parallel virtual machine is a freely available library for running processes on more than one processor on one or more machines ([Geist-1994]). EON can take advantage of PVM to distribute jobs over multiple processors. To do this PVM must be installed on all the machines EON will be distributed over. The PVM source is freely available from [http://www.csm.ornl.gov/pvm/pvm\\_home.html](http://www.csm.ornl.gov/pvm/pvm_home.html). However many Linux distributions, and some Unix versions, include PVM by default. EON is built with the current PVM version 3.4.5, but should also work with PVM version 3.4.4. EON does not support PVM under Windows.

To use EON with PVM you must do one of the following.

- Place a link to the `eon` executable in `$PVM_ROOT/bin/$PVM_ARCH`
- Define the environment variable **PVM\_PATH**, which points to the directory where the EON executable resides.

The environment variables **PVM\_ROOT** and **PVM\_ARCH** should be defined globally as part of the PVM installation. **PVM\_PATH** is generally a user-defined environment variable, and must be defined for all shells (*i.e.*, you can't just define it in the shell you're launching EON in).

**Note:** There is no specific slave executable. The executable distributed for this program serves as both a master and slave PVM program as well as a single processor version.

### 2.3.1 PVM hosts file

After PVM and EON are installed, each EON job run via PVM must be supplied a file (via `-pvmconf`) that describes the hosts and number of CPU's to use. Each line in this file includes the word **host** followed by the hostname of the slave followed by the number of CPU's to use on that slave.

To use 10 slaves on a linux cluster (where nodes are named *linux1*, *linux2*, etc.)

```
host linux1 2
host linux2 2
host linux3 2
host linux4 2
host linux5 2
```

To use 32 CPU's on a single-image system like an SGI Altix, a single line can be used:

```
host altix 32
```

## 2.4 Open MPI

EON supports MPI-1, or Message Passing Interface 1, on all platforms except Microsoft Windows, Solaris & AIX. MPI, like PVM, enables distributing EON runs over multiple machines and processors.

EON uses the Open MPI implementation of MPI, found at <http://www.open-mpi.org>. Every version of EON includes a full Open MPI install, so no additional software is needed.

There are two requirements to run EON under Open MPI.

- Every machine in the cluster must have EON installed.
- The path to the top level OpenEye bin directory must be in the **PATH** environment variable on all machines, and before any other locations that may contain MPI executables (orted, mpirun, etc).

**Note:** While the OpenEye MPI install must be first in the path, it will not conflict with other installs. OpenEye puts two MPI related scripts in the top level OpenEye bin directory. The first, `oempirun`, is a replacement for `mpirun`. Given their different names, there will not be conflict. The `orted` command in the OpenEye bin directory is a wrapper script around the `orted` daemon. If MPI was not run with `oempirun` and an OpenEye application, it will fall back to the next instance of `orted` in the path.

### 2.4.1 Using Open MPI

Running EON under Open MPI makes use of a wrapper script `oempirun` located in the OpenEye bin directory.

To run EON under Open MPI:

```
prompt@textgreater[] oempirun [Open MPI options] eon [EON Options]
```

Begin by generating a text file that will include the MPI hosts you plan to use and the number of processes on each (for this example, we'll call this file `hosts`). The file should contain a line for each machine with the name of the machine, a space, then `slots=N`, where N is the number of processors for your run. For this example, you would want the file to look like:

```
c1 slots=4
c2 slots=4
c3 slots=4
c4 slots=4
c5 slots=4
```

Now the following EON command-line will start a job with 1 master and 19 slaves.

```
prompt@textgreater[] oempirun -hostfile hosts eon [EON Options]
```

The master will be on `c1`, where the job is being started. All the i/o for the run will be from the master machine, and all results and logging information will be combined.

The command line may also be used to specify hosts. For example, to run 3 EON processes on the hosts `larry`, `curly` and `moe`:

```
prompt@textgreater[] oempirun -np 3 -host larry,curly,moe eon -query molecule.oeb \
-dbase database.oeb
```

The above command line will run 3 EON processes on the hosts larry, curly and moe. The master machine, which needs a license, would be larry. It is important to note that the eon passed to oempirun does not take a path.

## 2.4.2 Open MPI License

Most files in this release are marked with the copyrights of the organizations who have edited them. The copyrights below generally reflect members of the Open MPI core team who have contributed code to this release. The copyrights for code used under license from other parties are included in the corresponding files.

```
Copyright (c) 2004-2008 The Trustees of Indiana University and Indiana
                        University Research and Technology
                        Corporation. All rights reserved.
Copyright (c) 2004-2008 The University of Tennessee and The University
                        of Tennessee Research Foundation. All rights reserved.
Copyright (c) 2004-2008 High Performance Computing Center Stuttgart,
                        University of Stuttgart. All rights reserved.
Copyright (c) 2004-2007 The Regents of the University of California. All rights reserved.
Copyright (c) 2006-2008 Los Alamos National Security, LLC. All rights reserved.
Copyright (c) 2006-2008 Cisco Systems, Inc. All rights reserved.
Copyright (c) 2006-2008 Voltaire, Inc. All rights reserved.
Copyright (c) 2006-2008 Sandia National Laboratories. All rights reserved.
Copyright (c) 2006-2008 Sun Microsystems, Inc. All rights reserved.
                        Use is subject to license terms.
Copyright (c) 2006-2008 The University of Houston. All rights reserved.
Copyright (c) 2006-2008 Myricom, Inc. All rights reserved.
Copyright (c) 2007-2008 UT-Battelle, LLC. All rights reserved.
Copyright (c) 2007-2008 IBM Corporation. All rights reserved.
Copyright (c) 1998-2005 Forschungszentrum Juelich, Juelich Supercomputing
                        Centre, Federal Republic of Germany
Copyright (c) 2005-2008 ZIH, TU Dresden, Federal Republic of Germany
Copyright (c) 2007      Evergrid, Inc. All rights reserved.
Copyright (c) 2008      Institut National de Recherche en
                        Informatique. All rights reserved.
Copyright (c) 2007      Lawrence Livermore National Security, LLC.
                        All rights reserved.
Copyright (c) 2007-2008 Mellanox Technologies. All rights reserved.
Copyright (c) 2006      QLogic Corporation. All rights reserved.
```

Additional copyrights may follow

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright

notice, this list of conditions and the following disclaimer listed in this license in the documentation and/or other materials provided with the distribution.

- Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

The copyright holders provide no reassurances that the source code provided does not infringe any patent, copyright, or any other intellectual property rights of third parties. The copyright holders disclaim any liability to any recipient for claims brought against recipient by any third party for infringement of that parties intellectual property rights.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

# USAGE

## 3.1 Introduction

EON calculates the electrostatic similarity between two small molecules in the form of an Electrostatic Tanimoto (ET) score. Given a query molecule and a set of interesting molecules (ROCS overlay hits, for example), EON will calculate the Electrostatic Tanimoto between each database molecule and the query. Note that EON does not perform any overlay or alter the input orientation of the structures. They must be pre-aligned to the query on input. Also, since electrostatics calculations require high quality partial charges, EON will calculate new partial charges for the input structures using MMFF94. If the user provides an input file that contains structures with higher-quality partial charges, EON can use them as well.

EON is also dependent on pKa state and formal charges as these have a significant impact on electrostatics. EON now has the ability to adjust both the query and database molecule to a neutral pH model. This feature is on by default, but can be turned off by using appropriate command line flags.

Since electrostatics overlays are very dependent on alignment and require a good quality alignment between query and database molecule, ROCS provides the best input to EON. However, electrostatic complementarity is more dependent on subtle conformational changes than shape is, so there are several steps that can be taken to ensure the best possible success with EON.

Firstly, one can ensure that ROCS outputs multiple interesting conformers per molecule. ROCS includes a flag `-eon_input` that allows generation of a multi-conformer set of ROCS-aligned output specifically for input into EON. This file can be generated in parallel with a ROCS hit list so that in a single ROCS run you can find ROCS hits and prepare EON input. Please see the ROCS documentation for more detail on these flags.

Secondly, EON reads one or more conformers from the input file and uses technology from OMEGA to expand terminal torsions to search for subtle changes in conformation that might increase the score without changing the overall shape overlap with the query. To score just the input conformers and not search for alternate terminal conformations, the `-scoreonly` flag is provided.

Part of understanding EON results is visualization of the electrostatic grids used in the calculation. Although off by default, when writing EON results to a binary (OEB) file, ET grids can be attached to each molecule and visualized using the EON View mode in VIDA.

Since EON calculations can be time-consuming (approximately 1 molecule per second per CPU), EON can use the same distributed computing technology, PVM, that ROCS uses to help distribute the workload across a cluster of machines.

## 3.2 Command Line Interface

A description of the commandline interface can be obtained by executing EON with the `--help` option.

```
prompt@textgreater[] eon --help
```

will generate the following output:

Help functions:

```
eon --help simple      : Get a list of simple parameters
eon --help all         : Get a complete list of parameters
eon --help <parameter> : Get detailed help on a parameter
eon --help html        : Create an html help file for this program
```

### 3.2.1 Required Parameters

#### **-dbase <filename>**

File containing one or more 3D molecules to score against a reference or query molecule. If only this flag is given, EON will use the first molecule in the dbase file as the query and score all the remaining molecules against it. This is most useful when scoring ROCS results when ROCS was run with *-eon\_input* equal to **true** since the ROCS query and therefore EON query will be the first in the input file. The query or reference molecule can also be specified separately using the *-query* flag below.

File format for *-dbase* can be one of:

File type	Extension
OEBinary	.oeb .oeb.gz
SDF	.sdf .mol .sdf.gz .mol.gz
MOL2	.mol2 .mol2.gz
PDB	.pdb .ent .pdb.gz .ent.gz
MacroModel	.mmod .mmod.gz

### 3.2.2 Optional Parameters

#### Execute Options

##### **-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. EON generates a new parameter file containing the full set of execution parameters upon every execution. The name of the parameter file written by EON is created by combining the prefix base name with the *'parm'* extension.

##### **-pvmconf**

A text file specifying a PVM configuration. The *-pvmconf* option is only available on platforms where EON supports PVM. For every host in the cluster it should contain a line:

```
host host_name n
```

where n is the number of processors on the host.

## Input Options

### **-query <filename>**

File containing one 3D molecule to use as a query. File format can be any of the formats given in the table for *-dbase* above.

### **-charges**

Specifies charges to be used on the query. Default is to calculate them internally with *mmff*. The option *-charges existing* will use precalculated charges that must be set in the input files.

[default = *mmff*]

### **-scdbase**

Since EON reads multi-conformer input molecules, when reading from non-binary files, EON will compare consecutive molecules and if they are determined to be the same structure, they will be concatenated into a single, multi-conformer molecule. If the user desires to score each input conformation independently, then using this flag will turn off the conformer comparison step.

[default = *false*]

## Output Options

### **-offormat <extension>**

Format of output structure file(s). The default is *sdf* so that ET score can be included as tag data. The format for the file is determined by giving the extension. Valid values include all formats listed above for *-query*

[default = *sdf*]

### **-prefix <prefix>**

Defines a prefix used to name output files. Using *-prefix FOO* will create a hits structure file named like *FOO\_hits.sdf* and a report file, *FOO.rpt*

[default = *EON*]

### **-besthits <N>**

Process entire *dbase* file but only keep *N* best scores sorted by property specified by *-rankby*. Using a value of *0* implies no hitlist will be maintained and structures will all be scored and output in input order.

[default = *500*]

### **-maxhits <N>**

Stop after finding first *N* hits. This option overrides any setting for *-besthits*

[default=*0*]

### **-rankby <score>**

Property to use to sort hitlist. Values include *ET\_combo*, *ET\_pb* and *ET\_coul*.

[default = *ET\_combo*]

### **-cutoff <score>**

Minimum score to keep as a hit.

[default = *-1.0*]

### **-outputquery**

Write the query to the top of the hits file. This make visualization of results much easier inside *VIDA*.

### **-scoreonly**

Turn off the terminal torsion conformer search and just score each input conformation as-is.

**-hitsfile <filename>**

Explicit filename for writing hits. Overrides the default filename created from *-prefix*.

**-reportfile <filename>**

Explicit filename for writing hits. Overrides the default filename created from *-prefix*.

### Log Output Options

**-logfile <filename>**

Filename for log file. Overrides log filename created from *-prefix*.

**-progress**

Method for showing job progress on the command line. Choices include:

- percent - show a percent complete progress bar (DEFAULT)
- log - echo the log message for each molecule
- dots - show dots as in EON 1.1
- none - print nothing to console

[default = percent]

**-statusfile**

Write status info to this file. Use “none” for no status file.

**-verbose**

Give verbose output to console instead of simple progress.

[default = false]

### ZAP/PB Options

**-fixpka\_query**

Apply a neutral pH model to the query molecule.

[default = true]

**-fixpka\_dbase**

Apply a neutral pH model to the database molecules.

[default = true]

**-salt <F>**

Add salt to the Zap calculation. To aid in moderating large, local charges, salt is added into the calculation. Legal values are between 0.0 and 0.1 (mM).

[default = 0.04]

**-writegrid**

Write ET grid to output attached to each molecule. Useful for visualization in VIDA but this only works when writing hits to an OEB (.oeb or .oeb.gz) file. Note that while this feature is quite useful, grids do take a large amount of memory so care should be taken when using this feature for hit lists of more than 500 molecules.

[default = false]

## PVM

- pvmdebug**  
Generate an enormous volume of PVM debug information.
- pvmlog**  
Filename used for PVM log file
- pvmpass**  
Number of molecules to pass to a slave at one time

### 3.2.3 Example Commands

The simplest way to run EON is to use the `-eon_input` flag in ROCS to create a set of ROCS-aligned structures with the ROCS query at the top of the file. By default, if EON is only provided a dbase file, it will assume the first molecule is the query.

```
prompt@textgreater[] eon -dbase rocs_eon_input.oeb.gz
```

Or you can provide a single molecule in a query file and a set of molecules in a dbase file. A common example would be to use a ROCS query as the EON query and the ROCS hits file as the dbase file for EON.

So for example:

```
prompt@textgreater[] eon -dbase rocs_hits_1.sdf -query rocsquery.sdf
```

will score all the structures in `rocs_hits_1.sdf` against the molecule in `rocsquery.sdf` and place the structures in `EON_hits.sdf` (with ET scores in SD tag data) and a table of results in `EON.rpt`.

To keep only the best 100 hits, use:

```
prompt@textgreater[] eon -dbase rocs_hits_1.sdf -query rocsquery.sdf -besthits 100
```

Note that the use of an output structure file is mostly for generation of a single file containing structures and tag data such that loading this single file into VIDA provides easy analysis of the results. If however, the only real desire is for the numerical scores, use `-nostructs` to suppress the creation of an output structure file.

```
prompt@textgreater[] eon -dbase rocs_hits_1.sdf -query rocsquery.sdf -nostructs
```

By default, EON calculates partial charges using MMFF94. However, if you have structure files that already contain good partial charges in both your dbase file and query file, you can tell EON to use them instead:

```
prompt@textgreater[] eon -dbase rocs_hits_chgs.mol2 -dbase_charges file
-query rocsquery_chg.mol2 -query_charges file
```

To prevent continually over-writing output files, the `-prefix` flag allows you to give unique names to these files.

```
prompt@textgreater[] eon -dbase rocs_hits_1.sdf -query rocsquery.sdf -prefix FOO
```

will write the hit structures into a file named `FOO_hits.sdf` and the numerical values will be in `FOO.rpt`. The parameter file for this run will likewise be named `FOO.parm`.

To prevent EON for searching alternate terminal torsions, use the `-scoreonly` flag.

```
prompt@textgreater[] eon -dbase rocs_eon_input.oeb.gz -scoreonly
```

Finally, to create ET grids as used in the calculation and attach them to each output molecule. Note, this only works for OEB output and can be very memory intensive.

```
prompt@textgreater[] eon -dbase rocs_eon_input.oeb.gz -oformat .oeb.gz -writegrid
```

### 3.3 Report File

The EON report file format appears as a tab-delimited file with the following fields. Since the names of the query and the hits are of indeterminate length, fixed size fields for these names could result in loss of information. Unfortunately this gives a file that is hard to read in a terminal session, but it can easily be read into a spreadsheet program or into the data manager in VIDA.

**Name** This is the name of the database molecule.

**EONQuery** This is the name of the query molecule.

**Rank** The numerical ranking in the hitlist, based on the chosen score to rank by. Using the defaults, this is **ET\_combo**. Can be altered by using *-rankby* command line switches. If no hitlist was used in the calculation, this field will be 0 (zero).

**ET\_pb** This is the value of electrostatic Tanimoto, using full Poisson-Boltzmann (PB) electrostatics.

**ET\_coul** This is the value of electrostatic Tanimoto using only the coulombic part of PB electrostatics.

**ET\_combo** Sum of **ET\_pb** and **EON\_shape\_tani**. This is a useful score that takes into account both shape match and ET match.

**EON\_shape\_tani** This the shape Tanimoto between the given molecule and the query. For calculations that use *-shapeonly*, this will be the same as the output Tanimoto from ROCS. When EON is allowed to alter terminal torsion, this will give the final shape Tanimoto.

# THEORY

EON uses a field-based measure of Tanimoto to compare the electrostatic potential of two small molecules. This electrostatic potential is calculated internally using Zap, OpenEye's Poisson-Boltzman (PB) electrostatics toolkit.

The basic equation for a field Tanimoto is:

$$Tanimoto_{A,B} = \frac{\int A(\vec{r}) * B(\vec{r})}{\int A(\vec{r}) * A(\vec{r}) + \int B(\vec{r}) * B(\vec{r}) - \int A(\vec{r}) * B(\vec{r})}$$

The two boundary cases for Electrostatic Tanimoto occur when B = A:

$$\begin{aligned} Tanimoto &= \frac{\int A(\vec{r}) * A(\vec{r})}{\int A(\vec{r}) * A(\vec{r}) + \int A(\vec{r}) * A(\vec{r}) - \int A(\vec{r}) * A(\vec{r})} \\ &= 1 \end{aligned}$$

and the opposite case, when B = -A:

$$\begin{aligned} Tanimoto &= \frac{\int A(\vec{r}) * -A(\vec{r})}{\int A(\vec{r}) * A(\vec{r}) + \int -A(\vec{r}) * -A(\vec{r}) - \int A(\vec{r}) * -A(\vec{r})} \\ Tanimoto &= \frac{-\int A(\vec{r}) * A(\vec{r})}{\int A(\vec{r}) * A(\vec{r}) + \int A(\vec{r}) * A(\vec{r}) + \int A(\vec{r}) * A(\vec{r})} \\ &= -\frac{1}{3} \end{aligned}$$

In EON, we report two different Electrostatic Tanimoto(ET) measures, based on the outer dielectric used in the PB calculation. ET\_pb uses an outer dielectric of 80, while ET\_coul uses a value of 2.0. The rationale for using a PB electrostatic field is that the external potential is dampened by orientation of the aqueous solvent. It is a common observation that proteins essentially act to reproduce the aqueous desolvation of well-bound ligands. As a result a PB electrostatic field is more likely to correctly capture the essential elements of binding than that from the Coulombic field. However, this would still seem to be a point to be proven. As such we provide both Tanimotos. They typically track each other very closely.

For hit list ranking, we also report a score (ET\_combo) that is the sum of the Shape Tanimoto (ST) and the PB Electrostatic Tanimoto (ET\_pb).



# RELEASE NOTES

## 5.1 EON v2.1.0

*June 2011*

### 5.1.1 Enhancements

- The command line flags for choosing which charges to use have been simplified. There is a single flag `-charges`. The default is still to calculate MMFF charges. To use charges you have precalculated outside EON, first make sure you have charges for both the query and the dbase molecules, then use `-charges existing`.

**Note:** To use `existing` charges you must provide input files in a format that can store charges. This means input should be in either OEB or MOL2.

- This release adds EON to the set of applications that now use a common script in `openeye/bin` to determine the appropriate architecture and run the appropriate binary.
- On Windows, there is now an installer that installs the documentation and sets up a command prompt to facilitate running the EON command line.

### 5.1.2 Bug Fixes

- Fixed a bug to ensure that SD data present in the database is passed on to the EON hitlist. Note that this is specific to SD data other than ROCS scores. Since EON can modify the conformation, any ROCS scores would no longer be valid so they are removed from the output. Note that the score reported as `EON_ShapeTanimoto` is the final Shape Tanimoto calculated from the best conformation found inside EON.
- Fixed a bug that caused a crash when trying to write an empty hitlist.
- Fixed a bug that prevented `-salt` from working.
- Adding better error checking and messages for input. All input must be molecules and cannot be shape queries.

## 5.2 EON v2.0.0

*August 2007*

## 5.2.1 Enhancements

This is a significant new release of EON. This release incorporates many internal changes in how Electrostatic Tanimoto is calculated compared to EON 1.1.

- Salt in the PB calculation is now on by default to help moderate large, local charges. This should make EON 2.0 perform much better than 1.1 in the case of molecules with formal charges.
- OpenEye's neutral pH model has been incorporated allowing the setting of a pH model on both the query and database molecules. Since a significant use of EON is similarity searching, having a consistent model between the query and database molecules is very important. However, for the case where the states are well defined, this feature can be turned off.
- Instead of the slow and buggy `-spin` model for searching terminal torsions, EON 2.0 uses technology from OMEGA to only search reasonable torsions and to prevent creation of bad structures while searching for better electrostatic overlap.
- The hitlist is now sorted by default by a new score, `ET_combo`, which is the sum of the Shape Tanimoto (ST) and the Electrostatic Tanimoto (`ET_pb`).
- To make visualization of results easier, EON can write out the actual ET grids to the hits file, attached to each hit. This file can be loaded into VIDA 3.0 or later and the grids can be visually compared.
- There are also several changes to the mechanics of running the calculation including a new `-progress` feature and a more detailed PVM log output for PVM jobs.

---

## LIST OF SELECTED EON PUBLICATIONS

- S.W. Muchmore, A.J. Souers and I. Akritopoulou-Zanze, **The Use of Three-Dimensional Shape and Electrostatic Similarity Searching in the Identification of a Melanin-Concentrating Hormone Receptor 1 Antagonist**, *Chemical Biology & Drug Design*, Vol. 67(2), pp 174-176 **2006**
- Markt, P., Petersen, R.K., Flindt, E.N., Kristiansen, K., Kirchmair, J., Spitzer, G., Distino, S., Schuster, D., Wolber, G., Laggner, C. & Langer, T., **Discovery of Novel PPAR Ligands by a Virtual Screening Approach Based on Pharmacophore Modeling, 3D Shape and Electrostatic Similarity Screening**, *Journal of Medicinal Chemistry*, Vol. 51(20), pp. 6303-6317, **2008**.
- Naylor, E., Arredouani, A., Vasudevan, S.R., Lewis, A.M., Parkesh, R., Mizote, A., Rosen, D., Thomas, J.M., Izumi, M., Ganesan, A., Galione, A. & Churchill, G.C., **Identification of a Chemical Probe for NAADP by Virtual Screening**, *Nature Chemical Biology*, Vol. 5, pp. 220-226, **2009**.
- Tresadern, G., Bemporad, D. & Howe, T., **A Comparison of Ligand Based Virtual Screening Methods and Application to Corticotropin Releasing Factor 1 Receptor**, *Journal of Molecular Graphics and Modelling*, Vol. 27(8), pp. 860-870, **2009**.
- Zavodszky, M.I., Rohatgim, A., Van Voorst, J.R., Yan, H. & Kuhn, L.A., **Scoring Ligand Similarity in Structure-Based Virtual Screening**, *Journal of Molecular Recognition*, Vol. 22(4), pp. 280-292, **2009**.
- Lopez-Ramos, M. & Perruccio, F., **HPPD: Ligand- and Target-Based Virtual Screening on a Herbicide Target**, *Journal of Chemical Information and Modeling*, Vol. 50(5), pp 801-814, **2010**.



# BIBLIOGRAPHY

[Geist-1994] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. & Sundaram, V., **PVM - Parallel Virtual Machine: A User's Guide and Tutorial for Networked Parallel Computing**; *MIT Press*, 1994



# INDEX

## Symbols

-besthits <N>  
    eon command line option, 11

-charges  
    eon command line option, 11

-cutoff <score>  
    eon command line option, 11

-dbase <filename>  
    eon command line option, 10

-fixpka\_dbase  
    eon command line option, 12

-fixpka\_query  
    eon command line option, 12

-hitsfile <filename>  
    eon command line option, 11

-logfile <filename>  
    eon command line option, 12

-maxhits <N>  
    eon command line option, 11

-offormat <extension>  
    eon command line option, 11

-outputquery  
    eon command line option, 11

-param  
    eon command line option, 10

-prefix <prefix>  
    eon command line option, 11

-progress  
    eon command line option, 12

-pvmconf  
    eon command line option, 10

-pvmdebug  
    eon command line option, 13

-pvmlog  
    eon command line option, 13

-pvmpass  
    eon command line option, 13

-query <filename>  
    eon command line option, 11

-rankby <score>  
    eon command line option, 11

-reportfile <filename>  
    eon command line option, 12

-salt <F>  
    eon command line option, 12

-scdbase  
    eon command line option, 11

-scoreonly  
    eon command line option, 11

-statusfile  
    eon command line option, 12

-verbose  
    eon command line option, 12

-writegrid  
    eon command line option, 12

## A

APPNAME\_OE\_ARCH, 4

## E

environment variable

    APPNAME\_OE\_ARCH, 4  
    OE\_ARCH, 4  
    OE\_LICENSE, 3  
    PATH, 4–6  
    PVM\_ARCH, 5  
    PVM\_PATH, 5  
    PVM\_ROOT, 5

eon command line option

    -besthits <N>, 11  
    -charges, 11  
    -cutoff <score>, 11  
    -dbase <filename>, 10  
    -fixpka\_dbase, 12  
    -fixpka\_query, 12  
    -hitsfile <filename>, 11  
    -logfile <filename>, 12  
    -maxhits <N>, 11  
    -offormat <extension>, 11  
    -outputquery, 11  
    -param, 10  
    -prefix <prefix>, 11  
    -progress, 12

- pvmconf, 10
- pvmdebug, 13
- pvmlog, 13
- pvmpass, 13
- query <filename>, 11
- rankby <score>, 11
- reportfile <filename>, 12
- salt <F>, 12
- scdbase, 11
- scoreonly, 11
- statusfile, 12
- verbose, 12
- writegrid, 12

## O

- OE\_ARCH, 4
- OE\_LICENSE, 3

## P

- PATH, 4–6
- PVM\_ARCH, 5
- PVM\_PATH, 5
- PVM\_ROOT, 5