
Lexichem
Chemical Name to Structure and
Structure to Name Conversion

version 1.6

OpenEye Scientific Software, Inc.

July 2, 2007

9 Bisbee Ct, Suite D
Santa Fe, NM 87508
www.eyesopen.com
support@eyesopen.com

Copyright © 1997-2007 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 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. Alpha is a trademark of Digital Equipment Corporation. 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 MDL Information Systems, Inc. SMILES, SMARTS, and SMIRKS may be trademarks of Daylight Chemical Information Systems. Macromodel is a trademark of Schrödinger, Inc. Schrödinger, Inc may be a wholly owned subsidiary of the Columbia University, New York.

Python is a trademark of the Python Software Foundation.

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

"The forefront of chemoinformatics" is a trademark of Daylight Chemical Information Systems, Inc.

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

CONTENTS

1	Introduction	1
1.1	Input Name Representation	1
1.2	Output Name Representation	2
1.3	Output Name Styles	2
1.4	Examples of Name Style Differences	3
2	nam2mol	5
2.1	Command Line Interface	5
2.2	Command Line Options	6
3	mol2nam	7
3.1	Command Line Interface	7
3.2	Command Line Options	8
4	OEIUPAC Functions	10
4.1	OECapitalizeName	10
4.2	OECreateIUPACName	10
4.3	OEGetCIPStereo	11
4.4	OEIUPACIsLicensed	11
4.5	OENameLocant	12
4.6	OEParseIUPACName	12
4.7	OESetCIPStereo	12
4.8	OEToASCII	12
4.9	OEToBritish	12
4.10	OEToDutch	13
4.11	OEToEUCJP	13
4.12	OEToFrench	13
4.13	OEToGerman	13
4.14	OEToHTML	13
4.15	OEToItalian	14
4.16	OEToJapanese	14
4.17	OEToPolish	14

4.18	OEToSJIS	14
4.19	OEToSpanish	14
4.20	OEToSwedish	14
4.21	OEToUTF8	15
5	OEIUPAC Constants	16
5.1	OENamStyleAutoNom	16
5.2	OENamStyleCAS	16
5.3	OENamStyleCASIndex	16
5.4	OENamStyleIUPAC	16
5.5	OENamStyleIUPAC79	17
5.6	OENamStyleIUPAC93	17
5.7	OENamStyleOpenEye	17
5.8	OENamStyleSystematic	17
5.9	OENamStyleTraditional	17
A	Release Notes	18
A.1	June 2007, v1.6	18
A.2	September 2006, v1.5	19
A.3	February 2006, v1.4	19
A.4	January 2006, v1.3	20
A.5	October 2005, v1.2	20
A.6	August 2005, v1.1	20
A.7	June 2005, v1.0	21
A.8	January 2005, v1.0b3	21
A.9	August 2004, v1.0b2	21
A.10	March 2004, v1.0b1	24
B	Bibliography	25

Introduction

The OpenEye Lexichem product contains two applications, `nam2mol` and `mol2nam`.

The `nam2mol` program is an application for converting compound names into chemical structures. The program currently converts text files containing a single name per line, in either American or British English, into database of molecules, in a choice of file formats, include MDL SD file, SMILES, SLN or Tripos `.mol2`. This program does not require the input compound name to be the preferred IUPAC name of a compound, and it will work with a variety of traditional names and/or alternate IUPAC forms.

The `mol2nam` program is an application to perform the opposite conversion, translating chemical structures into a reasonable compound name.

1.1 Input Name Representation

The `oeiupac` library currently processes NUL (zero) terminated ASCII character strings, therefore greek characters, symbols, fonts and superscripts must be transliterated into the printable subset of ASCII.

When parsing compound names, the `oeiupac` library considers both spaces and tab characters as interchangeable, and any number of consecutive 'whitespace' characters are treated as a single space.

Currently, the name parsing is case insensitive, allowing arbitrary mixing of upper and lower case characters, *e.g.* initial letter capitalization.

Greek characters are understood in a number of different representations. The strings '\$a', '\$a}', 'alpha', '.alpha.', 'α', 'α' and 'α' are all understood to represent the greek character alpha, α .

There is no special representation for italic characters. Compound names such as '*tert*-butyl' and '*p*-aminobenzamide' are represented as 'tert-butyl' and 'p-aminobenzamide'. Both the long and short forms of prefixes can be used, allowing the above examples to also be written as 't-butyl' and 'para-aminobenzamide'.

1.2 Output Name Representation

Unrecognized functional groups, linkers or ring systems are denoted in the generated name as the string “BLAH”. As much of the name possible is generated resulting in compound names such as ‘dichloroBLAHcarboxylic acid’.

Generated compound names are entirely lower case, with no initial capitalization. Upper case characters are generated for locants and as described above, for BLAH.

When generating greek characters in compound names, the oeiupac library currently uses the dollar character followed by single letter representation. In this formalism, ‘\$a’ represents the greek character alpha, α , ‘\$b’ the greek character beta, β , ‘\$g’ the greek character gamma, γ and ‘\$l’ the greek character lambda, λ .

When generating superscripts, the oeiupac library currently uses the caret and curly braces representation. Hence ‘\$l⁵’ represents the greek character lambda followed by a superscript five, i.e. λ^5 . Similarly, ‘pentacyclo[4.2.0.0^{2,5}.0^{3,8}.0^{4,7}]octane’ would be the von Baeyer system name for cubane, i.e. pentacyclo[4.2.0.0^{2,5}.0^{3,8}.0^{4,7}]octane.

Multiple components in a disconnected molecule, apart from common salts and counter ions, are separated from each other by a semicolon followed by a space. Mixtures containing salts are written ordering the cations, before the compound name, followed by anions, finally followed by any common neutral molecules (e.g. hydrate or hydrochloride).

1.3 Output Name Styles

The Lexichem compound naming functionality supports the generation of several *styles* of compound name. The currently predefined name styles are OpenEye (the default), IUPAC, CAS, Traditional and Systematic.

OpenEye names loosely correspond to the kinds of names familiar to a medicinal chemist. These names are intended to be a subset of the IUPAC 2005 standard’s *acceptable* names, but not necessarily the PIN (Preferred IUPAC Name). These correspond to the types of names found in a Sigma-Aldrich catalogue or a Journal of Medicinal Chemistry article for example.

IUPAC names are intended to follow the IUPAC 2005 recommendations for the Preferred IUPAC Name (PIN). Unfortunately, this functionality is relatively recent, so the best that can be hoped for these names is that they are more IUPAC-like than the default OpenEye name style. Future release of Lexichem may further refine this definition to provide IUPAC2005, IUPAC93 and IUPAC79 name styles that reflect the corresponding standard’s preferred name.

The Lexichem CAS name style is intended to follow the Chemical Abstracts Service’s naming conventions, where they differ from IUPAC’s. Once again, as this functionality is relatively recent, the effect is to generate names that are more CAS-like than the default OpenEye name style.

The Traditional name style corresponds to forms of compound naming that are now no longer acceptable to the IUPAC rules. The boundary between whether a trivial/common name is considered OpenEye or

Traditional when it acceptable to IUPAC but not preferred is blurred, with OpenEye attempting to follow the more prevalent usage.

Finally, Systematic names correspond to the fully systematic IUPAC names that the IUPAC preferred names are slowly converging towards.

1.4 Examples of Name Style Differences

Some of the concepts explained in the previous section are probably best clarified through some real examples.

Example OpenEye vs. IUPAC vs. Systematic Differences

The SMILES string "O" is called "water" by the OpenEye name style, but "oxidane" by the IUPAC and Systematic name styles.

The SMILES "C#C" is called "acetylene" by the OpenEye and IUPAC name styles, but "ethyne" by the Systematic name style.

The SMILES prefix "*Nc1ccccc1" is called "anilino" by OpenEye and IUPAC, but "phenylamino" by systematic.

The SMILES prefix "*O[N+]#[C-]" is called "fulminato" by OpenEye, but "isocyanooxy" by IUPAC and Systematic.

The SMILES prefix "*C(=O)C" is called "acetyl" in OpenEye and IUPAC, but "ethanoyl" in Systematic.

The SMILES string "CC(=O)C" is called "acetone" in OpenEye, but "propan-2-one" in IUPAC and Systematic.

The SMILES string "C12C3C4C1C5C4C3C25" is called "cubane" in OpenEye, but is currently named "BLAH" in IUPAC and Systematic as we currently fail to name it as the preferred IUPAC2005 PIN: "pentacyclo[4.2.0.0^{2,5}.0^{3,8}.0^{4,7}]octane".

The SMILES string "C(=O)O" is called "formic acid" in OpenEye/IUPAC, but "methanoic acid" in Systematic.

Example OpenEye/IUPAC vs. CAS Differences

The SMILES string "c1ccccc1CCCCC" is named as "1-phenylheptane" by OpenEye and IUPAC, but as "heptylbenzene" by CAS.

The SMILES prefix "*[BH2]" is called "boranyl" by OpenEye and IUPAC, but as "boryl" by CAS.

Example OpenEye/IUPAC vs. Traditional Differences

The SMILES prefix `*S` is called “sulfanyl” by OpenEye and IUPAC, but as “mercapto” by Traditional.

The SMILES string `CCCCCCCC(=O)O` is called “nonanoic acid” by OpenEye and IUPAC, but as “pelargonic acid” by Traditional.

Nam2mol

OpenEye Scientific Software's `nam2mol` application converts compound names into molecular structures.

2.1 Command Line Interface

A description of the command line interface can be obtained by executing `nam2mol` with no arguments.

```
prompt> nam2mol
```

will generate the following output:

```
nam2mol v1.6  Name to Structure Conversion
OpenEye Scientific Software, June 2007

usage:  nam2mol [options] <namfile> [<outfile>]
        -empty: Output an empty molecule for unparseable names
        -depict: Generate 2D co-ordinates (if licensed)
        -tag: Set name as SD data with tag
```

The `nam2mol` program currently accepts no options, and takes one or two filenames of the command line. The first file on the command line is assumed to be compound name file in ASCII text format, and the optional second filename is treated as the output molecule file. A minus character may be used in place of the input filename to specify that the input is to be read from standard input, `stdin`, and in place of the output filename to specify that the output is to be written to standard output, `stdout`. If only one filename is specified on the command line, the output is written to `stdout` by default.

The file format of the output file is automatically determined from the file extension. The extensions `.smi`, `.can` and `.ism` may be used to specify SMILES format, `.sdf`, `.mdl` and `.mol` can be used to specify MDL connection table file formats, `.sln` for Sybyl line notation, `.mol2` for Tripos `.mol2` files, `.pdb` for PDB format files, etc...

2.2 Command Line Options

- empty** Write an empty connection table whenever unable to parse an input line. This function is useful for keeping track of which names were converted, as the output file is guaranteed to have the same number of connection tables as there were lines in the input file.
- depict** When writing output files that contain co-ordinates, this command line option can be used to generate suitable 2D co-ordinate depictions, provided that the OpenEye “ogham” toolkit is appropriately licensed.
- tag** Specify the tag field name to be used when writing MDL SD files. Normally, the original name is recorded in the title field of each output connection table. However, for MDL SD files the title is truncated to a maximum of 80 characters. This option allows the full name to additionally be written as a field to the file. For example, `-tag name` writes the name in the `<name>` data field.

Mol2nam

OpenEye Scientific Software's mol2nam application converts molecular structures into reasonable chemical names.

3.1 Command Line Interface

A description of the command line interface can be obtained by executing mol2nam with no arguments.

```
prompt> mol2nam
```

will generate the following output:

```
mol2nam v1.6  Structure to Name Conversion  
OpenEye Scientific Software, June 2007
```

```
usage:  mol2nam [options] <infile> [<outfile>]  
-iupac: Generate IUPAC200x-like names  
-cas:   Generate Chemical Abstracts CAS-like names  
-traditional: Generate traditional common names  
-systematic: Generate fully systematic names  
-delim: Append name to title with delimiter  
-tag:   Set name as SD data with tag  
  
-uk:   Generate British spellings  
-de:   Generate German compound names  
-es:   Generate Spanish compound names  
-fr:   Generate French compound names  
-it:   Generate Italian compound names  
-jp:   Generate Japanese compound names  
-nl:   Generate Dutch compound names  
-pl:   Generate Polish compound names  
-se:   Generate Swedish compound names  
  
-ascii: Encode using 7-bit ASCII  
-utf8:  Encode using Unicode UTF-8  
-html:  Encode using HTML
```

```
-sjis: Encode using Shift-JIS  
-eucjp: Encode using EUC-JP
```

Command line options are distinguished from filenames by having a '-' prefix. Options can appear anywhere on the command line, *i.e.* before, after or in between filenames. When incompatible options are specified, the last one given on the command line takes effect.

The first file on the command line is assumed to be the input molecule file in any of a number of popular connection table formats. If no output file is specified, the program writes a name per line, for each connection table in the input file to standard output, stdout. If a second filename is specified, it treated as the output molecule files, and each of the input molecules is written to it, with the title of each record set to the assigned name.

A minus character may be used in place of the input filename to specify that the input is to be read from standard input, stdin, and in place of the output filename to specify that the output is to be written to standard output, stdout.

The file format of the input file is automatically determined from the file extension. The extensions `.smi`, `.can` and `.ism` may be used to specify SMILES format, `.sdf`, `.mdl` and `.mol` can be used to specify MDL connection table file formats, `.sln` for Sybyl line notation, `.mol2` for Tripos `.mol2` files, `.pdb` for PDB format files, etc...

3.2 Command Line Options

- ascii** Encode the output using 7-bit ASCII, converting accented characters to their unaccented forms.
- autonom** Attempt to generate MDL/Beilstein AutoNom-like names. By default, mol2nam uses the OpenEye name style. MDL's AutoNom normally generates capitalized names, which can be controlled via the `-capitalize` command line option.
- capitalize** Capitalize the appropriate letter of the generated name.
- cas** Attempt to generate CAS-like names, as used by the Chemical Abstracts Service (CAS). By default, mol2nam uses the OpenEye name style.
- casidx** Attempt to generate CAS permuted index-like names, as used by the Chemical Abstracts Service (CAS). By default, mol2nam uses the OpenEye name style.
- de** Generate German compound names.
- delim** By default, the connection tables written to the output file have their title replaced with the generated compound name. However, if the `-delim` option is given followed by a delimiter string, the name is appended to the original title separated by the specified delimiter.
- es** Generate Spanish compound names.
- eucjp** Encode the output using EUC-JP to represent Japanese characters. This is normally used in conjunction with the `-jp` command line option.

- fr** Generate French compound names.
- html** Encode the output using HTML markup to represent greek characters, foreign characters and superscripts.
- it** Generate Italian compound names.
- iupac** Attempt to generate the Preferred IUPAC Name (PIN) of a compound as defined by the IUPAC200x standard. By default, mol2nam uses the OpenEye name style.
- iupac79** Attempt to generate an IUPAC 1979-style name. By default, mol2nam uses the OpenEye name style.
- iupac93** Attempt to generate an IUPAC 1993-style name. By default, mol2nam uses the OpenEye name style.
- jp** Generate Japanese compound names. See also the `-sjis` and `-eucjp` command line options.
- nl** Generate Dutch compound names.
- tag** When the output file is to be written in MDL SD file format, also write the compound name in the specified data tag.
- traditional** Attempt to generate traditional, common or archaic names for a compound. By default, mol2nam uses the OpenEye name style.
- pl** Generate Polish compound names.
- se** Generate Swedish compound names.
- sjis** Encode the output using Shift-JIS to represent Japanese characters. This is normally used in conjunction with the `-jp` command line option.
- systematic** Attempt to generate (fully) systematic names. By default, mol2nam uses the OpenEye name style.
- uk** Generate compound names with British spelling.
- utf8** Encode the output using UTF-8.

OEIUPAC Functions

This chapter describes the Lexichem C++ API. These functions reside in the `OEIUPAC` namespace, and operate on `OEChem` molecules.

4.1 OECapitalizeName

```
std::string OECapitalizeName(const char *ptr)
```

Capitalize the appropriate first letter of a name generated by `OECreateIUPACName`. This function should be called after translating the name to a foreign language, but before converting the character set encoding.

4.2 OECreateIUPACName

```
std::string OECreateIUPACName(const OEChem::OEMolBase &mol,  
                             const unsigned char *style=OENamStyleOpenEye)
```

This function attempts to generate a “reasonable” IUPAC name for the given molecule, `mol`, and return the result in a C++ STL string. These “reasonable” names attempts to be one of the recommended IUPAC names for a compound, however occasionally this function may fall back to using IUPAC “systematic” naming for parts of a molecule. Any parts of a molecule that cannot be named, result in the substring `BLAH` appearing in the returned string.

The optional `style` argument can be used to control and customize the *style* of the names generated by this function. The nine currently predefined name styles are `OENamStyleOpenEye` (the default), `OENamStyleIUPAC`, `OENamStyleIUPAC79`, `OENamStyleIUPAC93`, `OENamStyleTraditional`, `OENamStyleAutoNom`, `OENamStyleCAS`, `OENamStyleCASIndex` and `OENamStyleSystematic`.

After the name has been generated it may be translated into one of several languages, for example using the `OEToGerman` or `OEToJapanese` functions, then optionally capitalized using

OECapitalizeName, and finally converted into a final character encoding, for example using OEToUTF8 or OEToHTML.

4.3 OEGetCIPStereo

```
char OEGetCIPStereo(const OEChem::OEMolBase &mol,
                   const OEChem::OEAtomBase *atm)
char OEGetCIPStereo(const OEChem::OEMolBase &mol,
                   const OEChem::OEBondBase *bnd)
```

These functions return the Cahn-Ingold-Prelog descriptor for the given atom or bond stereo center, from the stereochemistry set on the OEMolBase. For chiral atom centers, the OEAtomBase form of this function returns either 'R' or 'S' for specified CIP stereo centers, 'N' for CIP stereo centers that don't have stereo specified (i.e. OEAtomBase::HasStereoSpecified returns false), and 'X' for atoms that are not CIP stereo centers. For double bonds, the OEBondBase form of this function returns either 'E' or 'Z' specified CIP stereo centers, 'N' for CIP stereo centers that don't have stereo specified (i.e. OEBondBase::HasStereoSpecified returns false), and 'X' for bonds that are not CIP stereo centers.

4.4 OEIUPACIsLicensed

```
bool OEIUPACIsLicensed(const char *features = 0, unsigned int *expdate = 0)
```

Determine whether a valid license file is present. This function may be called without a legitimate runtime license to determine whether it is safe to call any of OEIUPAC's functionality.

The "features" argument can be used to check for a valid license to OEIUPAC along with that feature. For example, to verify that OEIUPAC can be used from Python:

```
if (!OEIUPACIsLicensed("python"))
    OThrow.Warning("OEIUPAC is not licensed for the python feature");
```

The second argument can be used to get the expiration date of the license. This is an array of size three with the date returned as {day, month, year}. Even if the function returns false due to an expired license, the expdate will show that expiration date. A value of a zeroes implies that no license or date was found.

```
unsigned int expdate[3];
if (OEIUPACIsLicensed(0, expdate))
{
    OThrow.Info("License expires: day: %d month: %d year: %d",
               expdate[0], expdate[1], expdate[2]);
}
```

4.5 OENameLocant

```
std::string OENameLocant(unsigned int loc)
```

Generate the lexical form of a LexiChem locant index. The molecule's created by the function `OEParseIUPACName` are annotated with locant information in each atom's integer atom type field. These locant indices may be retrieved using `OEChem's OEAtomBase::GetIntType` method.

4.6 OEParseIUPACName

```
bool OEParseIUPACName(OEMolBase &mol, const char *name)
```

This function parses the compound name (not necessarily a systematic or preferred IUPAC name) from the NUL-terminated string given by `name`, and places the processed molecule in `mol`. This function returns `true` if the name could be parsed without problems. When returning `false`, the contents of `mol` contain as much of the name as could be processed.

4.7 OESetCIPStereo

```
bool OESetCIPStereo(OEChem::OEMolBase &mol,
                   OEChem::OEAtomBase *atm, char s)
bool OESetCIPStereo(OEChem::OEMolBase &mol,
                   OEChem::OEBondBase *bnd, char s)
```

These functions set the internal `OEChem` stereochemistry from the given CIP stereo descriptor. For the `OEAtomBase` form, the descriptor `s` must be either 'R' or 'S', and for the `OEBondBase` form, the descriptor `s` must be either 'E' or 'Z'. This function returns `true` if the stereochemistry was successfully set, and `false` otherwise: *i.e.* the descriptor was invalid or the specified atom or bond was not a CIP stereo center.

4.8 OEToASCII

```
std::string OEToASCII(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatIUPACName` post-processed by a language translation function, from the default ISO-8859-1 encoding, which includes 8-bit European characters, into a reduced 7-bit ASCII representation by stripping the accents off of the 8-bit characters.

4.9 OEToBritish

```
std::string OEToBritish(const char *ptr, bool sulph)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName`, from the default American output, to either British spelling (when `sulph` is `true`) or IUPAC international spelling (when `sulph` is `false`).

4.10 OEToDutch

```
std::string OEToDutch(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName`, from English to Dutch.

4.11 OEToEUCJP

```
std::string OEToEUCJP(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName` post-processed by a call to `OEToJapanese`, from the default encoding which uses `\u` escapes to represent unicode characters to instead use the EUC-JP character encoding for japanese characters.

4.12 OEToFrench

```
std::string OEToFrench(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName`, from English to French.

4.13 OEToGerman

```
std::string OEToGerman(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName`, from English to German.

4.14 OEToHTML

```
std::string OEToHTML(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName` possibly post-processed by a language translation function, from the default encoding to use HTML mark-up to represent accented characters, unicode characters and superscripts.

4.15 OEToItalian

```
std::string OEToItalian(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName`, from English to Italian.

4.16 OEToJapanese

```
std::string OEToJapanese(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName`, from English to Japanese.

4.17 OEToPolish

```
std::string OEToPolish(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName`, from English to Polish.

4.18 OEToSJIS

```
std::string OEToSJIS(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName` post-processed by a call to `OEToJapanese`, from the default encoding which uses `\u` escapes to represent unicode characters to instead use the Shift-JIS character encoding for japanese characters.

4.19 OEToSpanish

```
std::string OEToSpanish(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName`, from English to Spanish.

4.20 OEToSwedish

```
std::string OEToSwedish(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName`, from English to Swedish.

4.21 OEToUTF8

```
std::string OEToUTF8(const char *ptr)
```

Convert the string `ptr`, typically the output of the function `OECreatelUPACName` post-processed by a language translation function, from the default ISO-8859-1 encoding which uses `\u` escapes to represent unicode characters to instead use the UTF-8 character encoding for accented and japanese characters.

OEIUPAC Constants

5.1 OENamStyleAutoNom

```
const unsigned char OENamStyleAutoNom[];
```

This constant is used to specify the predefined MDL/Beilstein AutoNom name style to the `OECreatelUPACName` function.

5.2 OENamStyleCAS

```
const unsigned char OENamStyleCAS[];
```

This constant is used to specify the predefined Chemical Abstracts Service (CAS) name style to the `OECreatelUPACName` function.

5.3 OENamStyleCASIndex

```
const unsigned char OENamStyleCASIndex[];
```

This constant is used to specify the predefined Chemical Abstracts Service (CAS) permuted index name style to the `OECreatelUPACName` function.

5.4 OENamStyleIUPAC

```
const unsigned char OENamStyleIUPAC[];
```

This constant is used to specify the predefined IUPAC 200x name style to the `OECreatelUPACName` function.

5.5 OENamStyleIUPAC79

```
const unsigned char OENamStyleIUPAC79[];
```

This constant is used to specify the predefined IUPAC 1979 name style to the `OECreatIUPACName` function.

5.6 OENamStyleIUPAC93

```
const unsigned char OENamStyleIUPAC93[];
```

This constant is used to specify the predefined IUPAC 1993 name style to the `OECreatIUPACName` function.

5.7 OENamStyleOpenEye

```
const unsigned char OENamStyleOpenEye[];
```

This constant is used to specify the (default) predefined `OpenEye` name style to the `OECreatIUPACName` function.

5.8 OENamStyleSystematic

```
const unsigned char OENamStyleSystematic[];
```

This constant is used to specify the predefined systematic name style to the `OECreatIUPACName` function.

5.9 OENamStyleTraditional

```
const unsigned char OENamStyleTraditional[];
```

This constant is used to specify the predefined traditional name style to the `OECreatIUPACName` function.

Release Notes

A.1 June 2007, v1.6

On a benchmark of 250251 compounds in the NCI00 database, `mol2nam` is able to convert 233010 structures (93.11%) to names without BLAH. Of these 233010 names, `nam2mol` is able to convert 221331 (94.99%) back into structures.

This release includes a significant number of improvements to both name generation and name parsing. For example, both name generation and parsing now do a much better job on ring fusion nomenclature, for names like 5,6,7,8-tetrahydro[1,2,4]triazolo[4,3-a]pyridine. There's also much improved handling of charged ring systems. The name parsing conversion rate for the 71367 compound names in the 2003 Maybridge catalog is now 93.25% in v1.6, up from 80.80% in v1.5.

In name generation, new naming styles have been added for MDL/Beilstein AutoNom style names, for CAS permuted index style names (and there are new placeholder styles for IUPAC79 and IUPAC93 naming). A large number of improvements have been made to names generated using the "traditional" naming style. A new `OE CapitalizeName` API function is available to capitalizing the appropriate first letter of a generated name, such as `p-tert-Butylbenzoic acid`.

Several bug fixes have been made to the Cahn-Ingold-Prelog (CIP) chirality perception implementation.

The `OE ParseIUPACName` function is now able return supplementary locant annotations for each atom. This function now stores an integer locant code/identifier in the integer atom type field of each atom, which may be retrieved using the `OEChem::OEAtomBase::GetIntType` method and converted into a readable/displayable string using the recently exposed `OEIUPAC::OENamelocant` function. This functionality is a recent addition (obviously), and most but not all supported ring systems and parents have locant annotations in this initial release.

Finally, for the adventurous, new APIs for translating compound names from foreign languages into English are available as the experimental `OEFromJapanese`, `OEFromSwedish` and `OEFromSpanish` functions. Additionally, a `OEFromUTF8` function is available for converting UTF-8 encoded strings into the escaped sequences expected by these functions (effectively the inverse of `OETOUTF8`).

A.2 September 2006, v1.5

On a benchmark of 250251 compounds in the NCI00 database, `mol2nam` is able to convert 223066 structures (89.14%) to names without BLAH. Of these 223066 names, `nam2mol` is able to convert 192487 (86.29%) back into structures.

This release includes a significant number of improvements to both name generation and name parsing. For example, `nam2mol` now supports more numbered locants, such as “N1-methylaniline” and for ‘Maybridge-style’ locant names such as N’ 1 (interpreted as the more common N1’). These and similar changes have increased the conversion rate for the 71367 compound names in the 2003 Maybridge catalog, from 69.51% in v1.4 to 80.80% in v1.5.

This release includes the ability to generate compound names in Japanese, and much improved Spanish and Polish naming support. In order to better support internationalization, APIs are now available to map from the default ISO-8859-1 output to either 7-bit ASCII, UTF-8, HTML and for Japanese locales, Shift-JIS or EUC-JP.

Although impossible in the general case, several improvements have been made to Lexichem’s compound naming such that the assigned names are now more stable under arbitrary input ordering of atoms and bonds.

A.3 February 2006, v1.4

On a benchmark of 250251 compounds in the NCI00 database, `mol2nam` is able to convert 221254 structures (88.41%) to names without BLAH. Of these 221254 names, `nam2mol` is able to convert 192345 (86.93%) back into structures.

Lexichem v1.4 is predominantly a maintenance to provide a version of the `oeiupac` library that is compatible with OEChem v1.4. However, there have been a number of significant improvements to name parsing, and minor improvements to name generation since last month’s v1.3 release.

This release also includes the ability to generate compound names in several languages. In addition, to British spellings, Lexichem can now generate German, Italian, French, Spanish, Swedish, Dutch and Polish names. Whilst the translations for German, Italian, Swedish and Polish are quite comprehensive, those for French, Spanish and Dutch are less complete.

A potential ambiguity with the ring names “oxazole” and “thiazole” has also been resolved. The IUPAC documentation states that it is permissible to omit locants from Hantzsch-Widman names when the locants are consecutive, *i.e.* 1,2,3,4-tetrazole may be written as tetrazole, and 1,2-oxazirine is preferred as oxazirine. Unfortunately, this conflicts with the traditional interpretations of oxazole as meaning 1,3-oxazole and thiazole as 1,3-thiazole. Instead the traditional names `isoxazole` and `isothiazole` denote the 1,2- forms. This ambiguity, that affected IUPAC-style (but not OpenEye-style) names, has been resolved by preserving the locants, so that the IUPAC names 1,2-oxazole, 1,3-oxazole, 1,2-thiazole and 1,3-thiazole are now generated for `isoxazole`, `oxazole`, `isothiazole` and `thiazole` respectively.

A.4 January 2006, v1.3

On a benchmark of 250251 compounds in the NCI00 database, `mol2nam` is able to convert 221205 structures (88.39%) to names without BLAH. Of these 221205 names, `nam2mol` is able to convert 183444 (82.93%) back into structures.

The major announcement of this release is the support for stereochemistry in compound naming. The CIP rules for assigning R/S descriptors to tetrahedral chiral centers, and E/Z descriptors to double bonds are used during name generation.

A.5 October 2005, v1.2

On a benchmark of 250251 compounds in the NCI00 database, `mol2nam` is able to convert 220949 structures (88.29%) to names without BLAH. Of these 220949 names, `nam2mol` is able to convert 182438 (82.57%) back into structures.

A.6 August 2005, v1.1

On a benchmark of 250251 compounds in the NCI00 database, `mol2nam` is able to convert 220924 structures (88.28%) to names without BLAH. Of these 220924 names, `nam2mol` is able to convert 177145 (80.18%) back into structures.

A new `OEIUPACIsLicensed` API has been added so allow applications to check whether Lexichem's parsing and naming functionality can safely be used.

A.6.1 OEParseIUPACName Improvements

The Lexichem name parsing routines now handle a small number of structural abbreviations when parsing names. For example, it can now handle names like "3-CF3-5-NO2-benzoic acid". The usual improvements in name parsing, including more entries for common names in the Lexichem dictionary. Support for names containing multiple explicit hydrogen locants, such as "pyrimidine-2,4(1H,3H)-dione" and "2,4(1H,3H)-pyrimidinedione".

A.6.2 OECreatelUPACName Improvements

A serious bug that could cause a core dump when naming thioperoxoic acids has been fixed. The performance of compound naming has been improved. The usual improvements in the names generated (following the IUPAC standards more closely).

A.7 June 2005, v1.0

On a benchmark of 250251 compounds in the NCI00 database, `mol2nam` is able to convert 220922 structures (88.28%) to names without BLAH. Of these 220922 names, `nam2mol` is able to convert 177032 (80.13%) back into structures.

A.7.1 OEParseIUPACName Improvements

In addition to a great many other improvements to the name parsing code, the `lexichem` parser now contains an internal dictionary allowing the recognition of common non-systematic names, such as “ranitidine” and “zantac”.

A.7.2 OECreatelUPACName Improvements

In addition to a great many improvements to the name generation code, the `lexichem` naming functionality now allows the specification of a naming style, allowing the compound to be named in either a traditional, `OpenEye`, `IUPAC`, `CAS` or `systematic` naming style.

A.8 January 2005, v1.0b3

On a benchmark of 250251 compounds in the NCI00 database, `mol2nam` is able to convert 220854 structures (88.25%) to names without BLAH. Of these 220854 names, `nam2mol` is able to convert 144821 (65.57%) back into structures.

A.9 August 2004, v1.0b2

On a benchmark of 250251 compounds in the NCI00 database, `mol2nam` is able to convert 208378 structures (83.27%) to names without BLAH. Of these 208378 names, `nam2mol` is able to convert 139100 (66.75%) back into structures.

A.9.1 OEParseIUPACName Improvements

Support for parsing substituents (name fragments) by generating a wildcard atom in SMILES, for example, “methyl” returns `*C` and “hydroxy” returns `*O`. Adds support for parenthesized indicated hydrogens, e.g. “naphthalen-1(2H)-one”. Tweaks to support names, such as “9,10-anthracene-dicarboxylic acid”,

where the hyphen after the stem would cause the parser to expect a list of locants. Fixes to multipliers of “oxy” (and related) linkers, so that “dineopentyloxybenzene” is correctly considered two copies of the prefix “neopentoxy”. Added support for “...ic aldehyde” and “...ic acid aldehyde” as synonyms of “...aldehyde”. Added support for both “yl” and “oyl” variants of traditional acid stems, *i.e.* both “crintonoyl” and “crotonyl”, and both “acryloyl” and “acrylyl”. Adds support for “...amido” as a traditional linker, *e.g.* “propanamido”, “acrylamido” and “benzamido”. Added support for traditional bivalent imides, *e.g.* “succinimide”, including N-substituted forms, *e.g.* “N-iodosuccinimide”. Fixes parsing of salt multipliers, *e.g.* “benzene trihydrate”. Fixes nitrogenous linker processing, *e.g.* “N,N-dimethylsulfamoyl chloride”. Adds support for locants alpha, beta and gamma (as written in english), allowing us to handle “alpha,alpha,alpha-trichlorotoluene” and “beta-mercaptoethanol”. Improvements to handle unspecified locants, *e.g.* “chlorophenyl” and “pentachlorophenyl”. Support for substitutions on “benzoyl” and “benzamido” prefixes, *e.g.* “4-nitrobenzoyl chloride”. Handling of names using di, tri etc. incorrectly when bis, tris etc. should have been used, *e.g.* “2,4-di(t-butylperoxy)hexane”. Improvements to ring locant numbers on “benzophenone” and “benzidine”, and support for primed locants on ring system parents. Adds N and N' locants to the hydrazine stem, *e.g.* “4-(N',N'-dihydroxyhydrazino)benzoic acid”.

Treat “imidamide” as a synonym for “amidine”, and “hydrazinyl” as a synonym for “hydrazino”.

Adds support for the element “columbium” (more commonly known as “niobium”).

Adds support for the prefixes “keto” (synonym for “oxo”), “allophanoyl”, “hydantoyl”, “ureido”, “carbamido”, “lauryl”, “myrsityl”, “palmityl”, “stearyl”, “carbamimidoyl”, “sulfinamoyl”, “thiocarbamoyl”, “carbamothioyl”, “carbamoyl”, “guanyl”, “morpholino”, “oxycyano”, “sulfinyl”, “sulfonyl” and “fulminato”.

Added support for “acetonyl” and “phenacyl” as vinyl-like prefixes, including their “...idene” and “...idyne” variants.

Added support for the traditional stems “vanillic acid”, “isovanillic acid”, “syringic acid”, “arachidic acid”, “behenic acid”, “carboceric acid”, “cerinic acid”, “ceromelissic acid”, “ceroplastic acid”, “cerotic acid”, “daturic acid”, “enanthic acid”, “geddic acid”, “gheddic acid”, “japanic acid”, “lacceric acid”, “lignoceric acid”, “margaric acid”, “melissic acid”, “montanic acid”, “pelargonic acid”, “psylic acid”, “thapsic acid”, “brassylic acid” and “pyruvic acid”.

Added support for the stems “ketene”, “thioketene”, “vanillin”, “isovanillin”, “rhodanine”, “allophanic acid”, “hydantoic acid”, “picoline”, “borate” [BH₄-], “fulvene”, “isobutene”, “isoprene”, “alloxane”, “barbituric acid”, “hydantoin”, “cytosine”, “guanine”, “hydroxylamine” and the common amino acids.

Adds support for the ring systems “benzimidazole”, “benzoimidazole”, “benzothiophene”, “benzoxazole”, “benzooxazole”, “benzothiazole”, “benzotriazole”, “benzotrioxazole”, “pyrene”, “perylene”, “as-indacene”, “s-indacene”, “pyrrolizine” and “quinolizine”.

Added support for the ring suffix “carbonyl chloride” (and other acid halides).

Added support for the suffixes “thioketone”, “sulfide”, “nitrite”, “azanium”, “thial”, “diazonium”, “arsonic acid”, “peroxoic acid”, “carboperoxoic acid”, “carbothioamide”, “carboximidamide”, “carboxamidine”, “aldehyde oxime” “one oxime”.

Added support for the linkers “mercuri”, “carbamimidoyl”, “sulfinamoyl”, “sulfamoyl”, “thiocarbamoyl”, “carbamothioyl”, “carbonimidoyl”, and “thioyl” (as in “ethanethioylbenzene”).

Added support for the salts “hydrobromide”, “hydrofluoride”, “hydroiodide”, “triiodide”, “hydrotriiodide”,

“sulfite”, “peroxide”, “perchlorate”, “perbromate”, “periodate”, “hydrate”, “nitrite”, “hypochlorite”, “chlorite”, “chlorate”, “bromate”, “iodate”, “nitrate”, “cyanide” and “cyanate” (including iso and thio variants).

Added support for “hydrochloric acid”, “hydrobromic acid”, “hydrofluoric acid”, “hydroiodic acid”, “hydrotriiodic acid”, and “tetric acid”.

A.9.2 OECreatelUPACName Improvements

Added support for spiro, bicyclo and large simple heterocycle naming, all with hetero replacement nomenclature. No longer elide ring system prefix locants, which fixes the ambiguity with “N-tetlinylacetamide” which should be “N-tetralin-1-ylacetamide”. Improvements to indicated hydrogen perception (a nitrogen with three ring bonds doesn’t require an indicated hydrogen), improving our naming of “indolizine”, “pyrrolizine” and “quionlazine”. Improvements to naming cycloalkane rings attached via an oxygen linker, *i.e.* “cyclopropoxy” instead of “cylopropyloxy”. Improvements in the handling of substituted linkers, “carbamoyl”, “thiocarbamoyl”, “carbamimidoyl”, “sulfamoyl” and “sulfinamoyl”. Improvements to alkyl chain termination, for example “carboxymethyl” instead of “2-hydroxy-2-oxoethyl” and “cyanomethyl” instead of “nitridoethyl”. Improvements to amide, thioamide and sulfonamide atom typing to allow $*C(=O)N=C*$ to be considered an N-substituted amide. Added support for more phosphane variants; $*=P-*$, $*=PH$ and $*\#P$. Add “oxamide” as a contraction of “oxalamide”.

Fixes in multiple suffix processing, *e.g.* “cyclohexane-1,4-diamine”. Fixes to the prefixes “acryloyl” and “proprioyl” (and their acid halides) which were previously incorrectly named “acrylyl” and “propioyl”. Fixed handling of atom typed metals, fixing “trichloromagnesium”. Corrected names for bivalent acid halides, *e.g.* “malonyl dichloride”. Made “benzoyl chloride” the preferred name for “benzenecarbonyl chloride” and likewise for other benzoic acid halides. Fix spelling typos in “acenaphthene” and “isothiazolidine”.

Added support for the parents “heptalene”, “octalene”, “hydroxylamine”, “hexahydropyrimidine” (formerly “1,2-diazinane”), and “hexahydropyridazine” (formerly “1,3-diazinane”).

Added support for the prefixes “acetonyl”, “aceonylidene”, “phenacyl”, “phenacylidene”, “ureido”, “anilino”, “hydantoyl”, “allophanoyl”, “amidino”, “acetoxyl”, “isopropoxy”, “isobutoxy”, “sec-butoxy”, “tert-butoxy”, “morpholino” (formerly “morpholin-4-yl”), “oxycyano”, “sulfinyl” ($*=S=O$), “sulfonyl” ($*=S(=O)=O$), and “methylene” (formerly “methylidene”).

Added support for the linkers “carbonimidoyl”, “mercuri” and “benzoyl” (replacing “phenylcarbonyl”).

Add support for the suffixes “amidine”, “ohydrazide”, “carbohydrazide”, “...one oxime”, “...al oxime”, “aldehyde oxime” and “nitrile oxide” (note we consider both $*C\#N=O$ and $*C\#[N+][O-]$ nitrile oxides).

Added support for the salts “triiodide”, “hydrotriiodide”, “hydrogen triiodide”, “perbromate”, “periodate”, “sulfide”, “sulfite”, “peroxide”, “iodate”, “nitrite” and “hypochlorite”.

Added support for the molecule “fulvene”.

A.10 March 2004, v1.0b1

On a benchmark of 250,251 compounds in the NCI00 database, `mol2nam` is able to convert 201004 structures (80.32%) to names without BLAH. Of these 201004 names, `nam2mol` is able to convert 111442 (55.44%) back into structures.

Bibliography

1. J. Brecher, "Name=Struct: A Practical Approach to the Sorry State of Real-Life Chemical Nomenclature", *Journal of Chemical Information and Computer Sciences (JCICS)*, Vol. 39, pp. 943-950, 1999.
2. R.S. Cahn, C.K. Ingold and V. Prelog, "Specification of Molecular Chirality", *Angew. Chem. Int. Ed. Engl.*, Vol. 5, pp. 385-414, 1966. Errata Vol. 5, p. 511, 1966.
3. Gernot A. Eller, "Improving the Quality of Published Chemical Names with Nomenclature Software", *Molecules*, Vol. 11, pp. 915-928, 2006.
4. Robert B. Fox and Warren H. Powell, "Nomenclature of Organic Compounds: Principles and Practice", Oxford University Press publishers, 2001.
5. L. Goebels, A.J. Lawson and J.L. Wisniewski, "AUTONOM: System for Computer Translation of Structural Diagrams into IUPAC-Compatible Names: 2. Nomenclature of Chains and Rings", *Journal of Chemical Information and Computer Sciences (JCICS)*, Vol. 31, pp. 216-225, 1991.
6. D. Hellwinkel, "Systematic Nomenclature of Organic Chemistry: A Directory to Comprehension and Application of its Basic Principles", Springer-Verlag publishers, 2001.
7. Paul Labute, "An Efficient Algorithm for the Determination of Topological RS Chirality", *Journal of the Chemical Computing Group*, On-line, November 1996.
8. John R. Levine, Tom Mason and Doug Brown, "lex & yacc", 2nd Edition, UNIX Programming Tools, O'Reilly and Associates publishers, 1992.
9. W.R. Peterson, "Formulacion Y Nomenclatura Quimica Organica", 15th Edition, EDUNSA publishers, Barcelona, 1993. (Spanish).
10. Polskie Towarzystwo Chemiczne, "Nomenklatura Zwiaskow Organicznych", Panstwowe Wydawnictwo Naukowe publishers, Warsaw, 1992. (Polish).
11. Polskie Towarzystwo Chemiczne, "Przewodnik Do Nomenklatury Zwiaskow Organicznych", Narodowy Komitet Miedzynarodowej Unii Chemii Czystej I Stosowanej publishers, Warsaw, 1994. (Polish).
12. V. Prelog and G. Helmchen, "Basic Principles of the CIP-System and Proposals for a Revision", *Angew. Chem. Int. Ed. Engl.*, Vol. 21, pp. 567-583, 1982.

13. K.J. Thurlow, "Chemical Nomenclature", Kluwer Academic Publishers, 1998.
14. Susanne Wikman, "Organisk-Kemisk Nomenklatur", Studentlitteratur Publishers, Lund, 2004. (Swedish).
15. J.L. Wisniewski, "AUTONOM: System for Computer Translation of Structural Diagrams into IUPAC-Compatible Names: 1. General Design", *Journal of Chemical Information and Computer Sciences (JCICS)*, Vol. 30, pp. 324-332, 1990.