**IUPAC International Chemical Identifier (InChI)**

**InChI version 1, Software version 1.06 pre-release**

**Release Notes**

Last revision date: May 18, 2020

This document is a part of the pre-release of the IUPAC International Chemical Identifier with InChIKey, version 1, software version 1.06.

CONTENTS

[Summary 3](#_Toc40699796)

[Security-related fixes 5](#_Toc40699797)

[New and updated features 5](#_Toc40699798)

[Support of pseudoelement (“Zz”) atoms 5](#_Toc40699799)

[Modifications in support of regular single-strand polymers 5](#_Toc40699800)

[InChI API 5](#_Toc40699801)

[Intel TBB scalable memory allocator support 6](#_Toc40699802)

[Outline 6](#_Toc40699803)

[How to use 7](#_Toc40699804)

[Other 9](#_Toc40699805)

[Structure perception option LooseTSACheck 9](#_Toc40699806)

[Changes to inchi-1 executable 10](#_Toc40699807)

[Testing 12](#_Toc40699808)

[Regression testing 12](#_Toc40699809)

[InChI round-trip test 13](#_Toc40699810)

[Known issues 14](#_Toc40699811)

[Distribution package 14](#_Toc40699812)

[Binaries 14](#_Toc40699813)

[Source codes and demo programs 15](#_Toc40699814)

[Other 17](#_Toc40699815)

[Acknowledgements 17](#_Toc40699816)

## Summary

This document introduces the pre-release of InChI Software version 1.06 (Spring 2020).

The current version of InChI Identifier is 1; the current stable version of the InChI software is 1.05 which is planned to be replaced with v. 1.06.

InChI Software v. 1.06 includes several additions and modifucations to previous versions; it is a combination of “bugfix release” and “feature release”.

The main novel features are as follow:

* support for pseudoelement (“Zz”) atoms;
* experimental support of InChI/InChIKey for regular single-strand polymers was implemented;
* experimental support of large molecules containing up to 32767 atoms was added;
* ability to read input files in Molfile V3000 format was added; it is necessary for treating large molecules (previous versions supported only V2000 format limited to not more than 1000 atoms). Provisional support for extended features of Molfile V3000 was also added;
* InChI API Library was significantly updated, in particular:

a novel API procedure for direct conversion of Molfile input to InChI has been added, as well as a demo program illustrating its use;

a whole new set of API procedures for both low and high-level operations (InChI extensible interface, IXA) has been added, as well as a demo program illustrating its use;

* the source code was significantly changed in order to ensure multi-thread execution safety of the InChI Library; also, several minor bugfixes/changes (including refactoring) were made to the sources;
* several convenience options were added to the inchi-1 executable;
* documentation was updated.

## Security-related fixes

…

## New and updated features

### Support of pseudoelement (“Zz”) atoms

**TODO: UPDATE**

### Modifications in support of regular single-strand polymers

**TODO: UPDATE**

### InChI API

**TODO: UPDATE**

InChI Software v. 1.06 includes modified Library, INCH API. The related changes are outlined below.

Added:

- API call IXA\_INCHIBUILDER\_SetOption\_Timeout\_Milliseconds()

- IXA\_MOL\_GetBondOtherAtom() exposed in the public API, as Paul Thiessen suggested

- API call IXA\_MOL\_ReserveSpace() (necessary in new mode IXA\_USES\_SMART\_ALLOCS)

- More IXA API functionality for polymers (API calls IXA\_MOL\_CreatePolymerUnit (), IXA\_MOL\_SetPolymerUnit (), IXA\_MOL\_GetPolymerUnitId(), IXA\_MOL\_GetPolymerUnitIndex() )

- Improved performance of IXA memory allocations by using expandable arrays

(per request by Daniel Lowe; see #ifdef IXA\_USES\_SMART\_ALLOCS)

- Made return codes of GetInChIFrom...() functions in case of InChI read error corresponding to what is declared in API description

InChI “old” API “modularized” version (that using INCHIGEN object) is now frozen and considered obsolete. It still works but will not receive further development.

For the details, see the document “InChI\_API\_Reference.pdf”, included in this distribution.

### Intel TBB scalable memory allocator support

InChI Software v. 1.06 provides optional support of Intel(R) Threading Building Blocks (TBB) scalable memory allocators (SMA) which may improve performance of InChI library in multi-threading program environment.

#### Outline

Upon using InChI library in multi-threading environment, memory allocation can become a serious performance bottleneck due to locking issues. As threads may compete for a global lock related to a single global heap, program’s behavior is not scalable and speed may even degrade if number of processor cores increases. Scalable memory allocators resolve the issues by relying on more sophisticated data structures/logics.

Intel TBB is free software package available for both Windows and Linux, licensed under the Apache License, Version 2.0, see file LICENSE in TBB sub-directory of distribution (there is also a commercial counter-part; consult <https://www.threadingbuildingblocks.org>).

TBB offers a “proxy” method to automatically replace C functions for dynamic memory allocation with its own scalable memory allocators to largely avoid contention (there exist other, non-automatic ways of using TBB allocators, which we do not touch).

This method may optionally be used with InChI Software library and, in some cases, may provide performance gain (but your mileage may vary, so try and check).

For additional info on Intel scalable memory allocator and Intel TBB in general, please visit <https://www.threadingbuildingblocks.org/documentation>   
or consult print literature, e.g., open access book “Pro TBB. C++ Parallel Programming with Threading Building Blocks” by Michael Voss, Rafael Asenjo, and James Reinders:  
<https://link.springer.com/book/10.1007/978-1-4842-4398-5>

#### How to use

Use SMA-aware versions of libinchi dynamic libraries (libinchi.dll under Windows and libinchi.so under Linux), either supplied with this distribution or compiled from sources.

To run the application calling libinchi, ensure that it has access to Intel TBB SMA dynamic library.

Under Linux, the easiest way is to load SMA libraries at application load time using the LD\_PRELOAD environment variable (another option is by linking the main executable file with the proxy library). As loader must be able to find the proxy library and the allocator library, one may include the directory with libraries in the LD\_LIBRARY\_PATH environment variable or add it to /etc/ld.so.conf.

Example:

export LD\_PRELOAD="/opt/intel/compilers\_and\_libraries\_2020.1.217/linux/tbb/lib/intel64\_lin/gcc4.8/libtbbmalloc\_proxy.so.2 /opt/intel/compilers\_and\_libraries\_2020.1.217/linux/tbb/lib/intel64\_lin/gcc4.8/libtbbmalloc.so.2"

./mol2inchi chembl\_26.sdf -Threads:4 -Perthread:1024 -NoWarnings -AuxNone -Key >/dev/null 2>m2i-4-TBB.log

Under Windows, the simplest way is to copy tbbmalloc\_proxy.dll and tbbmalloc.dll to the directory containing program and libinchi.dll. Alternatively, you may put directory containing SMA library(-ies) to PATH environment variable. Note that the libraries for 32-bit and 64-bit Windows are different, so be careful to properly match “bitness” of involved binaries.

Precompiled SMA-awareInChI library binaries are supplied in this distribution in directories:

XXX TODO: insert

XXX TODO: insert

Intel TBB SMA library binaries are also supplied, for convenience. They are located in

XXX TODO: insert

XXX TODO: insert

These libraries may also be downloaded from Intel TBB site.

Alternative to using precompiled InChI library SMA-enabled binaries, you may build them.

For that purpose, uncomment the line

#define USE\_TBB\_MALLOC 1

in file mode.h and build libinchi.dll under Windows as usual, or build libinchi.so under Linux using supplied makefile makefile\_with\_tbb.

### Other

#### Structure perception option LooseTSACheck

A new InChI structure-perception (i.e., available for generation of Std

InChI) option LooseTSACheck relaxes the strictness of tetrahedral stereo ambiguity check, for in-ring atoms, thus getting stereo descriptor instead

of undefined mark ‘?’.

It accounts for now common case of large molecules/large cycles, drawn and then automatically 'cleaned' by chemoinformatics software, see the example below.

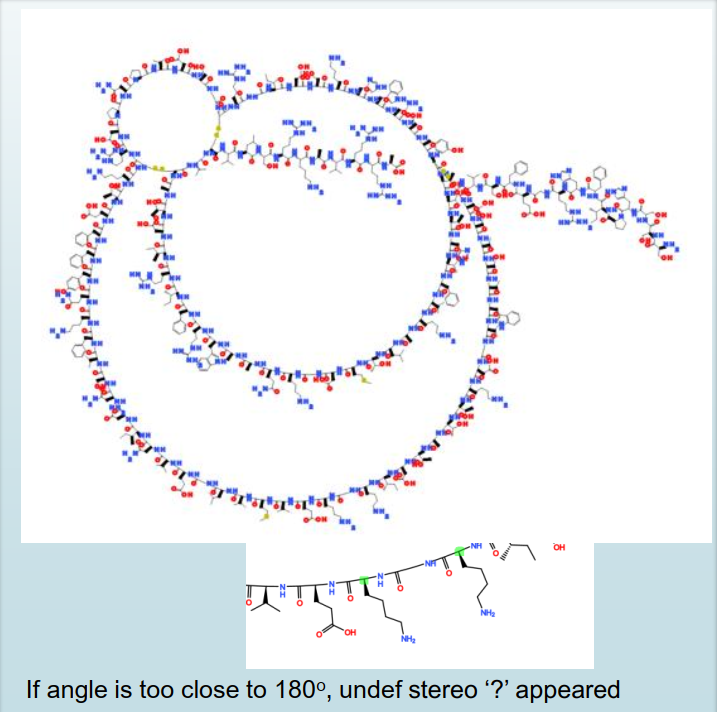


Fig. XXX. Auto-cleaned by drawing software large-ring molecule may get undefined stereo ‘?’ mark(s) due to too strict InChI’s tetrahedral stereo ambiguity check. Option LooseTSACheck fixes this issue and produces definite stereo descriptor.

#### Changes to inchi-1 executable

**TODO: UPDATE**

Several convenience changes were made to inchi-1 executable.

##### LargeMolecules

A new option ‘LargeMolecules’ instructs the program to accept molecules containing more than 1024 (but less than 32767) atoms.

##### Polymers105

A new option ‘Polymers’ instructs the program to treat polymer data as in v. 1.05 (no explicit Zz atoms).

##### Output at Error an empty InChI

The new option ‘Output at Error an empty InChI”, OutErrInChI (/OutErrInChI under Windows, -OutErrInChI under Linux) instructs the program to output empty InChI and corresponding InChIKey if error occurs (default behaviour is output nothing).

The Standard InChI and InChIKey for empty entity are:

InChI=1S//

InChIKey=MOSFIJXAXDLOML-UHFFFAOYSA-N

The Non-standard InChI and InChIKey are:

InChI=1//

InChIKey=MOSFIJXAXDLOML-UHFFFAOYNA-N

##### Selection of record in SDF input file

Option Record:N (/Record:N under Windows, -Record:N under Linux) instructs the program to process only the N-th record of the input file in SDF format.

This convenience option is equivalent to combined options Start:N (start processing from N-th record) and End:N (end processing at N-th record).

##### Other

Implementation of wildcard expansion in Allow Multiple Input, AMI mode (like in inchi-1 /AMI \*.mol) under Windows is improved, in order to tolerate possible large expansion volumes.

### Testing

**TODO: UPDATE**

#### Regression testing

There were several minor fixes/changes made after software release v. 1.05. To ensure compatibility with previous release, the new software has been extensively regression-tested against standard and non-standard InChIs generated with v. 1.04 Software, in both Windows and Linux environments, with the various option combinations for standard and non-standard InChI (totalling 33 option sets; options new in v. 1.05 were intentionally omitted).

The test sets included:

1. "InChI\_TestSet" (public). This is a test set of 2,186 structures which has been created previously and included in InChI Software distribution as "InChI validation suite".   The structures include some very tricky and "chemically strange" ones, to verify InChI behaviour in exotic cases.

2. “MDB” (proprietary) 100,000 structures.

3. "MSL-NIST" (proprietary). 191,436 structures.

4. "NCI" (public).    249,081 structures from "NCI Open Database Compounds", retrieved from: <http://cactus.nci.nih.gov/ncidb2/download.html>

5. “ChEMBL20” (public). 1,456,020 structures from http://www.ebi.ac.uk/chembl - the version of ChEMBL is chembl\_20. Retrieved on 2015-02-02.

6. "PubChem Compound" (public). 60,915,175 structures. Retrieved from PubChem on 2015-10-28.

No problems have been detected.

#### InChI round-trip test

**TODO: UPDATE**

The success rate for conversion of InChI to structures was measured using Rtrip option of test\_ixa program or two-pass (structure to InChI followed by InChI to structure) execution of inchi-1. In this round trip, InChIs generated from Molfile-format records were then used to restore structures and re-create InChIs from those structures, via corresponding API calls; original and final InChIs were then compared.

The results for ChEMBL v. 20 (1,456,020) and PubChem Compound (60,915,175 molecules; checked in 64 bit Linux environmemt) are as follow:

|  |  |
| --- | --- |
| Dataset | Success rate, % |
| Standard InChI | |
| ChEMBL | 99.98 |
| PubChem Compound | 99.93 |
| Non-standard (FixedH) | |
| ChEMBL | 99.90 |
| PubChem Compound | 99.93 |
| Non-standard (RecMet) | |
| ChEMBL | 99.98 |
| PubChem Compound | 99.92 |
| Non-standard (FixedH RecMet) | |
| ChEMBL | 99.90 |
| PubChem Compound | 99.92 |

## Known issues

**TODO: UPDATE**

Renumbering tests performed to date involved generation of InChI for 100 random re-numberings of atomic systems for ~6x104 PDB molecules.

Two entries were found to give different InChIs on renumbering:

PDB codes pdb2pfo and pdb4ged.

## Distribution package

**TODO: UPDATE**

### Binaries

This package includes 'command line' InChI executable and InChI API Library binaries (32 and 64 bit versions are supplied for both Windows and Linux).

Also included is winchi-1.exe, a graphical Windows application (a 32 bit version which will also run under 64 bit Windows).

**TODO: UPDATE**

File/directory INCHI-1-BIN

windows/

winchi-1.exe InChI graphical Windows application

windows/32bit/

inchi-1.exe InChI stand-alone command line

executable, 32 bit

windows/32bit/dll/

libinchi.dll InChI dynamic-link library, 32 bit

windows/64bit/

inchi-1.exe InChI stand-alone command line

executable, 64 bit

windows/64bit/dll

libinchi.dll InChI dynamic-link library, 64 bit

linux/

linux/32bit/

inchi-1.gz InChI stand-alone command line executable, 32 bit; gzipped

linux/32bit/so/

libinchi.so.1.05.00.gz shared library for InChI API, 32 bit; gzipped

linux/64bit/

inchi-1.gz InChI stand-alone command line

executable, 64 bit; gzipped

linux/64bit/so

libinchi.so.1.05.00.gz shared library for InChI API, 64

bit; gzipped

Note that InChI stand-alone executable inchi-1[.exe] does not require dll/so libraries.

### Source codes and demo programs

**TODO: UPDATE**

InChI Software binaries are placed in the file/directory INCHI-1-BIN. Example data files are placed in the file/directory INCHI-1-TEST. Documentation is placed in the file/directory INCHI-1-DOC.

InChI Software source codes are placed in the file/directory INCHI-1-SRC. This file/directory also contains examples of InChI API usage, for C ('inchi\_main', ‘mol2inchi’, ‘test\_ixa’, see projects for MS Visual Studio 2008 in 'vc9' and for gcc:Linux in ‘gcc’ subdirs) and Python 3 ('python\_sample'). Also supplied are InChI API Library source codes and related projects/makefiles.

The projects/makefiles necessary to build inchi-1 executable and demo programs are located within corresponding directories, as well as the source codes specific for these apps (see below). Note that a part of code which forms a common codebase is placed in special directory, INCHI-1-SRC/INCHI\_BASE/src. To ensure proper build of InChI applications/library, the directory structure below (a tree under INCHI-1-SRC) should be preserved. Note also that, though InChI library ‘libinchi’ may be build using its own projects/makefiles (under INCHI-1-SRC/INCHI\_API/libinchi/), it is automatically co-created upon building of any demo program.

INCHI-1-SRC/INCHI\_BASE

src C source files - common codebase

used to build both InChI Library

and inchi-1 executable

INCHI-1-SRC/INCHI\_EXE

bin directory where the binaries

of inchi-1 executable

are created/stored

inchi-1 a home directory for inchi-1

command-line executable

inchi-1/src C source files specific for

inchi-1 executable

inchi-1/gcc gcc makefiles for inchi-1

executable (Linux, 64- and 32-bit)

inchi-1/vc9 MS VS2008 project for inchi-1

executable (Windows)

INCHI-1-SRC/INCHI\_API

bin directory where the binaries

(of both library and all demo

programs) are created/stored

libinchi a home for InChI Software Library

libinchi/src C source files specific for

InChI Software Library

libinchi/gcc gcc makefiles for libinchi

library (Linux so)

libinchi/vc9 MS VS2008 project for libinchi

library (Windows dll)

demos a home for demo programs calling

InChI library (API)

demos/inchi\_main/src C source files specific for

inchi\_main demo

demos/inchi\_main/gcc gcc makefiles for inchi\_main

demo program (Linux)

demos/inchi\_main/vc9 MS VS2008 project for inchi\_main

demo program (Windows)

demos/mol2inchi/src C source files specific for

mol2inchi demo program

demos/mol2inchi/gcc gcc makefiles for mol2inchi

demo program (Linux)

demos/mol2inchi/vc9 MS VS2008 project for mol2inchi

demo program (Windows)

demos/test\_ixa/src C source files specific for

test\_ixa demo program

demos/test\_ixa /gcc gcc makefiles for test\_ixa

demo program (Linux)

demos/test\_ixa /vc9 MS VS2008 project for test\_ixa

demo program (Windows)

demos/python\_sample Python 3 source files specific for

Python demo program

For details, please refer also to ‘readme.txt’ files in the directories.

### Other

**TODO: UPDATE**

The documentation (Release Notes; InChI Technical Manual; InChI User Guide; InChI API Reference) in PDF format is supplied in the INCHI-1-DOC section of this distribution package.

Test data are supplied in the INCHI-1-TEST file/directory. Note the file pex.sdf there which contains Molfile presentation of the above-described example polymer data.

## Acknowledgements

We are grateful to many people and organizations who are continuously supporting InChI development, in various forms. Special thanks are due to those who specifically contributed to the current release/testing.