IUPAC International Chemical Identifier (InChI)
InChI version 1, software version 1.03 (2010)
Release Notes

Last revision date: June 15, 2010

This document is a part of the release of the IUPAC International Chemical Identifier with InChIKey, version 1, software version 1.03 (http://www.iupac.org/inchi).

InChI software v. 1.03 has merged functionality: it allows one to produce both standard and non-standard InChI identifiers, as well as their hashed representation (InChIKey).

CONTENTS

This document summarizes and briefly explains the new features of InChI software v. 1.03 (2010).

The current version of the Identifier is 1; the current version of the InChI software is 1.03 (June 2010). Previously released versions 1.01 (2006), 1.02-beta (2007), and 1.02-standard (2009), as well as all earlier versions, are now considered obsolete.

InChI generation

Standard vs. non-standard InChI generation

InChI software v. 1.03 (2010) has merged functionality: it allows one to produce both standard and non-standard InChI identifiers, as well as their hashed representation (InChIKey).

Standard InChI is marked with the prefix "InChI=1S/".

The prefix for non-standard InChI is simply "InChI=1/".

By default, InChI programs generate standard InChI/standard InChIKey. In particular, standard identifiers are generated when the software is used without any command-line options.

If some command-line options are specified, and at least one of them qualifies as related to non-standard InChI (see below), the executable produces non-standard InChI/InChIKey.

The same is true for general-purpose InChI API calls. For example, an API function `GetINCHI()` produces standard InChI by default and a non-standard one if some "InChI creation option" is specified in input parameters.

However, for compatibility with the previous v. 1.02-standard (2009) release, API calls which deal only with standard InChI – for example, `GetStdINCHI()` - are retained (technically, they provide pre-customized interface to general-purpose API functions).

Different markup for unknown and undefined stereo; new option "SLUUD"

In InChI software versions prior to 1.02-standard (2009), the two different signs were used to mark "unknown" and "undefined" stereo, 'u' and '?', resp. (Briefly: "undefined" means not given one while "unknown" means explicitly marked as unknown, e.g., with "wavy" bonds). In standard InChI software v. 1.02, the both signs were merged to '?' (that is, "unknown" stereo treated as "undefined").

In InChI software version 1.03 (2010), the same '?' sign for "unknown/undefined" is always used in standard InChI. It is also used in non-standard InChI, by default. However, it is possible to return to old policy of different 'u' and '?' signs.

For this purpose, the new option "/SLUUD" ("-SLUUD" under Linux) is introduced (SLUUD stands for 'Stereo labels for "unknown" and "undefined" are different').

Advanced tautomerism options (experimental)

The two options "KET" and "15T" provide, as an experiment, an access to advanced tautomerism detection. Option "/KET" ("-KET" under Linux) makes InChI account for keto-enol tautomerism; "/15T" ("-15T" under Linux) - for 1,5-tautomerism.

Note that the both kinds of advanced tautomerism detection are, actually, an extension to InChI Identifier v. 1. Their activation may affect very significant amount of InChI strings. Also, these features are not yet tested completely. Therefore, the both options are provided only for experimentation purposes and are strongly not recommended in routinely usage.

Structure perception and InChI creation options

Options affecting generation of InChI are divided on "structure perception" options and "InChI creation" options.

The perception options are considered drawing style/edit flags which affect the input structure interpretation and are not memorized. It is assumed that user may deliberately use these options to account for the specific features of structure collections. Whence, perception options may be used while generating standard InChI without a loss of its "standardness".

Perception options are listed in the following table.

Table 1. Structure perception options.

Structure perception option	Meaning	Default behavior (standard; if no option supplied)
NEWPSOFF	Both ends of wedge point to stereocenters	Only a narrow end of wedge points to stereocenter
DoNotAddH	All hydrogens in input structure are explicit	Add H according to usual valences
SNon	Ignore stereo	Use absolute stereo

There are several options (Table 2) which modify the interpretation of input stereochemical data. In principle, they also may be considered "structure perception" options. However, as the standard InChI, by definition, requires the use of absolute stereo (or no stereo at all), these "perception" options assume generation of non-standard InChI.

Table 2. Stereo interpretation options (lead to generation of non-standard InChI).

Structure perception option	Meaning	Default behavior (standard; if no option supplied)
SRel /SRac	Use relative/racemic stereo	Use absolute stereo
SUCF	Use Chiral Flag in MOL/SD file record: if On – use Absolute stereo, Off – Relative	Use absolute stereo (or another one if requested by SRel /SRac/SNon switches)

The creation options affects InChI algorithm, not a structure perception. They modify the defaults which are specified for standard InChI and significantly affect the final appearance (e.g., additional InChI layers may appear). Whence, using any of creation options qualifies the resulting identifier as a non-standard one.

Creation options used for generation of particular non-standard InChI may be appended to the created identifier, see below.

InChI creation options are listed in the following table.

Table 3. InChI creation options.

InChI creation option	Meaning	Default behavior (if no option supplied)
SUU	Always indicate unknown/undefined stereo	Does not indicate unknown/undefined stereo unless at least one defined stereo is present
SLUUD	Stereo labels for "unknown" and "undefined" are different, 'u' and '?', resp. (new option; see explanation)	Stereo labels for "unknown" and "undefined" are the same ('?')
RecMet	Include reconnected metals results	Do not include
FixedH	Include Fixed H layer	Do not include
KET	Account for keto-enol tautomerism (experimental; extension to InChI 1)	Ignore keto-enol tautomerism
15T	Account for 1,5-tautomerism (experimental; extension to InChI 1)	Ignore 1,5-tautomerism

Note that standard InChI is always generated if no InChI creation/stereo modification options are specified. Inversely, if any of SUU | SLUUD | RecMet | FixedH | Ket | 15T | SRel | SRac | SUCF options are specified in the command line, generated InChI will be non-standard one.

Saving InChI creation options; new option "SaveOpt"

The new command-line option "/SaveOpt" ("-SaveOpt" under Linux) allows one to append non-standard InChI string with saved InChI creation options.

The “SaveOpt appendix” currently consists of the two capital Latin letters which are separated from InChI string by backslash ‘\’.

Note that this appendix is not considered as an integral part (layer) of InChI itself; rather, it is an optional complement. It may or may not present after the end of InChI string (by default – no “SaveOpt” option – it is absent). To signify this, the appendix is separated from the previous sequence of symbols by a character which may not appear in any other place, a backslash.

Note also that InChI generation option “/SaveOpt” (and saved-options appendix) is not available for standard InChI as the latter is always created with the same options.

As for the encoding of saved options, the first SaveOpt letter encodes whether RecMet/FixedH/SUU/SLUUD switches were activated. Each of them is a binary switch ON/OFF, so it totals to 2*2*2*2=16 values which are encoded by capital letters ‘A’ through ‘P’.

The second letter encodes experimental (InChI 1 extension) options KET and 15T. Each of these options is a binary switch ON/OFF, so there are 2*2=4 combinations, encoded by ‘A’ through ‘D’. Note that anything but 'A' here would indicate "extended" InChI 1. Note that here is some reservation for future needs: the 2nd memo char may accommodate two more ON/OFF binary options (at 26-base encoding).

The exact encoding scheme is specified in the tables below.

Table 4. Meaning of the 1st SaveOpt letter.

Letter	RecMet	FixedH	SUU	SLUUD
A	OFF	OFF	OFF	OFF
B	OFF	OFF	OFF	ON
C	OFF	OFF	ON	OFF
D	OFF	OFF	ON	ON
E	OFF	ON	OFF	OFF
F	OFF	ON	OFF	ON
G	OFF	ON	ON	OFF
H	OFF	ON	ON	ON
I	ON	OFF	OFF	OFF
J	ON	OFF	OFF	ON

K	ON	OFF	ON	OFF
L	ON	OFF	ON	ON
M	ON	ON	OFF	OFF
N	ON	ON	OFF	ON
O	ON	ON	ON	OFF
P	ON	ON	ON	ON

Table 5. Meaning of the 2nd SaveOpt letter.

Letter	**Ket**	**15T**
A	OFF	OFF
B	OFF	ON
C	ON	OFF
D	ON	ON

Examples:

InChI=1/C9H11NO2.Na/c1-3-5(7(3)9(10)12)6-4(2)8(6)11;/h5-6,11H,1-2H3,(H2,10,12);/q;+1/p-1/t5?,6?;/i/hD/fC9H10NO2.Na/h11h,10H2;/q-1;m/i10D;\OA

(this identifier was created with options /RecMet /FixedH /SUU and /SaveOpt)

InChI=1/C9H11NO2.Na/c1-3-5(7(3)9(10)12)6-4(2)8(6)11;/h5-6,11H,1-2H3,(H2,10,12);/q;+1/p-1/t5?,6?;/i/hD\KA

(this identifier was created for the same input structure with options /RecMet /SUU and /SaveOpt)

InChI=1S/C9H11NO2.Na/c1-3-5(7(3)9(10)12)6-4(2)8(6)11;/h5-6,11H,1-2H3,(H2,10,12);/q;+1/p-1/i/hD

(this identifier was created for the same input structure with no InChI creation options)

InChIKey and hashing

(No) bugfix for InChIKey encoding

On Fall 2009, there was a considerable interest to InChIKey collision resistance which followed the report from Jonathan Goodman on the collisions of InChIKey 2nd block which were observed for stereo isomers of Spongistatin I (http://www.google.com/search?hl=en&source=hp&q=ICXJVZHDZFXYQC).

It should be explicitly stated that the collisions _must_ appear for this molecule having 24 tetrahedral stereo centers and 2 stereogenic double bonds – that is, by far beyond the capacity of InChIKey's 2nd block hash. Due to the very essence of hash functions, collisions are unavoidable in sufficiently large collections. Anyway, during the related inspection, a minor bug was found in the encoding of hashed InChI, which, in principle, may unfavorably affect the number of collisions.

The description of bug is as follows. SHA-256 algorithm produces 256-bit signature which is then truncated and encoded into the letters forming InChIKey. For the 2nd block, this presumes retaining the 37 most significant bits, from 0 to 36, of full 256 bits. Due to a bug, the layout of actually retained bits is slightly different: 0-31 and 36-39. (Analogously, for the 1st block the truncation procedure actually retained bits 0-63 and 71 instead of 0-64).

So a full exploration of Spongistatin I stereo isomers and corresponding InChIKey's was performed (by generating stereo isomers and computing their InChIKey's). It was found that for various subsets of full isomer set, the observed numbers of collisions N perfectly correspond to theoretical estimates. In the course of preparation of current software v. 1.03 release, the additional numerical experiments were performed. Namely, the bug in hash encoding was fixed and the numbers of collisions obtained for different (sub)sets of isomers were compared (i) to analogous numbers observed for original, non-fixed, version, and (ii) to theoretical estimates. The results are shown below.

Table 6. Stereo isomers of Spongistatin I: observed average numbers of non-unique InChIKey's vs. theoretical estimate for number of collisions (doublets). For the observed values the number of samplings used for averaging is given in parentheses.

Number of isomers in dataset	**Number of non-unique keys, original/no bugfix**	**Number of non-unique keys, bug-fixed**	**Theor. number of collisions (doublets)**
50,000	0.006 (500)	0.010 (500)	0.009
100,000	0.024 (250)	0.024 (250)	0.036
250,000	0.13 (100)	0.28 (100)	0.23
370,000	0.51 (100)	0.45 (100)	0.50
500,000	0.90 (100)	0.95 (100)	0.91
1,000,000	3.6 (50)	3.2 (50)	3.6
2,000,000	14.4 (50)	13.8 (50)	14.6
3,000,000	33.1 (50)	31.5 (50)	32.7
4,000,000	59.2 (50)	56.3 (50)	58.2
8,000,000	234.2 (50)	229.2 (50)	232.8

16,000,000	928.9 (40)	926.8 (40)	931.3
32,000,000	3753.1 (30)	3708.1 (30)	3725.3
67,108,864 (full set of 2^26 isomers)	16565*	16456**	16384

* All collisions are double except for 2 triplets

** All collisions are double except for 1 triplet

To enlarge a base for comparison, analogous numerical experiments were performed with the dataset of the same size, 2^26, and its subsets – but populated with generated stereo _and_ isotopic isomers of Spongistatin I. The results are shown below.

Table 7. Stereo/isotopo isomers of Spongistatin I: observed average numbers of non-unique InChIKey's vs. theoretical estimate for number of collisions (doublets). For the observed values the number of samplings used for averaging is given in parentheses.

Number of isomers in dataset	**Number of non-unique keys, original/no bugfix**	**Number of non-unique keys, bug-fixed**	**Theor. number of collisions (doublets)**
50,000	0.016 (500)	0.008 (500)	0.009
100,000	0.064 (250)	0.032 (250)	0.036
250,000	0.29 (100)	0.23 (100)	0.23
370,000	0.56 (100)	0.42 (100)	0.50
500,000	0.96 (100)	0.93 (100)	0.91
1,000,000	3.7 (50)	3.9 (50)	3.6
2,000,000	14.9 (50)	15.2 (50)	14.6
3,000,000	33.2 (50)	34.2 (50)	32.7
4,000,000	58.0 (50)	60.4 (50)	58.2
8,000,000	231.2 (50)	235.5 (50)	232.8
16,000,000	930.5 (40)	947.4 (40)	931.3
32,000,000	3702.1 (30)	3811.2 (30)	3725.3
67,108,864	16328*	16458**	16384

* All collisions are double except for 2 triplets

** All collisions are double except for 4 triplets

All in all, these additional experiments led to the same conclusion: as concerns collision resistance, InChIKey behaves nearly in theoretical manner, even without a bugfix. Moreover, one can not definitely state that bug-fixed version performs consistently better than an original one. Given this conclusion and the fact that the original version InChIKey's are already wide-spread, it was decided that the bugfix is not activated in InChI v. 1.03 (2010) software release.

Extended InChI hash

As there were concerns regarding the limited capacity of (2nd block of) InChIKey, a possibility to output the rest of 256-bit SHA-2 signature for 1st and 2nd blocks is introduced.

This is done with the new command-line options "/XHash1" and "/XHash2" ("-XHash1" and "-XHash2"under Linux).

Note that the rest of signatures appeared in hexadecimal notation to avoid confusion with InChIKey (which consists solely of capital English letters).

Example:

InChI=1S/C4H8/c1-3-4-2/h3-4H,1-2H3/b4-3+

InChIKey=IAQRGUVFOMOMEM-ONEGZZNKSA-N

XHash1=82ff0307735072b4ec27b9c093e9486dca09e8df1d0812c9

XHash2=403ee94266e1d8d96d47b99c4b17ff5f92e3a74e3f0f5ab8bc2775bb

InChI2InChI conversion and SaveOpt letters

A specific to InChI2InChI conversion issue is a consistent treatment of standard and non-standard identifiers, as well as SaveOpt letters. The following rules are implemented.

1. Generating standard InChI strings out of standard ones is disabled.

To ensure standard InChI validity, it may be generated only from a structure, not from another InChI.

2. Conversion of standard InChI string to a non-standard one is allowed.

If conversion explicitly requires /SaveOpt, the resulting InChI string will be appended with SaveOpt letters.

3. If source non-standard InChI is appended with SaveOpt letters (after '\') and conversion explicitly requires /SaveOpt, the resulting InChI string will be appended with SaveOpt letters. If a particular conversion option requires generally impossible transformation, accounting for the content of source SaveOpt (e.g., source indicates that input InChI has been created with no metal reconnection option and conversion requires /RecMet), then 1) warning is issued; 2) conflicting conversion option is ignored; 3) conflicting option will not affect SaveOpt letters of output string.

4. If source non-standard InChI is appended with SaveOpt letters (after '\') and conversion does not require /SaveOpt, the resulting InChI string will not be appended with SaveOpt letters. If a particular conversion option requires generally impossible transformation, accounting for the content of source SaveOpt (e.g., source indicates that input InChI has been created with no metal reconnection option and conversion requires /RecMet), then 1) warning is issued; 2) conflicting conversion option is ignored.

5. If source non-standard InChI is appended with SaveOpt letters (after '\') and conversion does not require /SaveOpt, "straight" conversion will be performed.

6. If source non-standard InChI string does not contain SaveOpt and conversion requires /SaveOpt, then 1) warning is issued; 2) /SaveOpt option is ignored; 3) "straight" conversion is performed and the resulting InChI string is not appended with SaveOpt letters.

Distribution package

Binaries

Included in this package are the following binaries.

File/directory INCHI-1-BIN

Folder win32

winchi-1.exe	InChI Windows program (version 1)
inchi-1.exe	InChI command line program (version 1)

Folder linux/32bit

inchi-1.gz	Linux i386 executable of inchi-1

Folder linux/64bit

inchi-1.gz	Linux amd64 executable of inchi-1

File/directory INCHI-1-API

Folder gcc_so_makefile/result

inchi_main	(Linux) the InChI software library demo application
libinchi.so.1.03.00	(Linux) the shared object

Folder vc9/inchi_dll/Release

inchi_main.exe	(Windows) the InChI software library demo application
libinchi.dll	(Windows) dynamic link library

InChI API/Library

Most notably, InChI API now allows one to generate both standard and non-standard InChI/InChIKey's. For example, an API function `GetINCHI()` produces standard InChI by default and a non-standard one if some "InChI creation option" is specified in input parameters. However, for compatibility with the previous v. 1.02-standard release, API calls which deal only with standard InChI – for example, `GetStdINCHI()` - are retained (technically, they provide pre-customized interface to general-purpose API functions).

The InChI API calls are documented in the separate "InChI API Reference Sheet" document and source code header file "inchi_api.h".

License and related

Minor technical clarification is made in references to GNU Lesser General Public License, LGPL. Referenced now is a specific version 2.1 of this license, http://www.opensource.org/licenses/lgpl-2.1.php (version 3.0 seems to differ significantly).

CML support/source code

Since 2004, InChI software has a support for CML (Chemical Markup Language). It is provided by CML reader written – and kindly supplied to NIST/InChI - by Peter Murray-Rust and Simon (Billy) Tyrrell, Unilever Centre for Molecular Sciences Informatics, University of Cambridge, UK. Some minor modifications were then made at NIST.

As CML reader has been developed outside InChI team, its source code was never distributed with InChI software package. Instead, pre-compiled binaries of InChI executables with built-in CML support were distributed.

Recently, after an interest expressed by users of InChI-discuss mailing list, P. Murray-Rust kindly provided permission to publish source code of CML reader with InChI package, as a free software under the same LGPL license as InChI itself. The sources are now included into the distribution package, and each file has a header suggested by Dr. P. Murray-Rust.

Testing

InChI generation/testing

The new software has been extensively regression-tested against standard InChI v. 1.02-standard (2009) and non-standard InChI v. 1.02-beta (2007) in both Windows and Linux environments.

The test sets included:

1. "InChI-101" (public). This is a test set of 2,186 structures which has been created previously and included into software v. 1.01 distribution as "InChI validation suite". The structures include some very tricky and "chemically strange" ones, to verify InChI behaviour in exotic cases.

2. "NCI" (public). 249,081 structures from "NCI Open Database Compounds", retrieved from:

   http://cactus.nci.nih.gov/ncidb2/download.html

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

4. "MDB" (proprietary). 100,000 structures.

5. "ChemSpider" (proprietary). 17,257,766 structures. ChemSpider database, as of December 2007, courtesy of A. Williams.

6. "PubChem Compound" (public). 38,966,897 structures. Retreived from PubChem on 2009-09-01.

7. On case-by-case basis, "PubChem Substance" (public) collection was used. This is useful for tests with "unpolished" structures: in contrast with "PubChem Compound", this collection (of 71,271,452 records) contains non-normalized structures just as they were deposited to PubChem. Retreived from PubChem on 2009-08-11.

All the above mentioned input files were in MDL SD format. Additionally, there were several test input files in CML format. The various option combinations were used in the tests.

Above described is related to regression testing of InChI generation. That is, tests assume the generation of InChI strings (from structures in SD file) and comparing them with InChI's obtained with "previous-state" software.

Additionally, we tested InChI2InChI conversion and InChI2Struct conversion performed by current software.

The last case (InChI2Struct) is related to "round-trip" testing:

[Source structure =>] InChI1 => Restored Structure => InChI2

(here successful test means that InChI2 generated from a restored structure does match InChI1; note that InChI2Struct conversion supposed that the structure is restored from InChI, not from AuxInfo)

The conversion success rate (executable: inchi-1) is summarized below.

Table 8. Conversion statistics.

Dataset	**No. of InChI's**	**InChI options***	**Conversion failure**	**Conversion mismatch**	**Total problems, %**	**Success rate, %**
InChI2InChI conversion						
PubChem Compound	38966897	Std	0	0	0	100
		Non-std-1	0	0	0	100
		Non-std-2	0	327	0.0008	99.999
		Non-std-3	0	0	0	100

		Non-std-4	0	327	0.0008	99.999
		Non-std-5	0	331	0.0009	99.999
PubChem Substance	60681047	Std	0	0	0	100
InChI2Struct conversion						
PubChem Compound	38966897	Std	7	18359	0.05	99.95
		Non-std-4	51	26428	0.07	99.93
		Non-std-5	51	27221	0.07	99.93
PubChem Substance	60681047	Std	19	61306	0.10	99.90

* Std	standard InChI
Non-std-1	/SUU /SLUUD
Non-std-2	/SUU /SLUUD /FixedH
Non-std-3	/SUU /SLUUD /RecMet
Non-std-4	/SUU /SLUUD /RecMet /FixedH
Non-std-5	/SUU /SLUUD /RecMet /FixedH /KET /15T

Bugfixes impact

There were several minor fixes after software release v. 1.02-standard (2009). In particular:
(1) the bug in normalization procedure for some structures (containing N2(+) fragment) which may result in producing different InChI strings for the same molecule, depending on original order of the atomic numbers (reported by Timo Boehme on March 2010: http://sourceforge.net/mailarchive/message.php?msg_name=4B98AD34.5000307%40ontochem.com) was fixed;

(2) the fix is implemented for minor issues (reported by Hinnerk Rey) related to normalization for structures containing positively charged tetra-coordinated phosphorus (or sulfur) connected to the negatively charged oxygen;
(3) the bug in treating stereochemistry of allenes with non-carbon substituents (reported by Burt Leland) was fixed.

The combined impact of bugfixes is rather small: as estimated using PubChem Substance collection, only 300 standard InChI strings out of 60,681,047 - that is, $5*10^{-4}$ % - are affected.