IUPAC International Chemical Identifier (InChI)

InChI version 1, Software version 1.06 pre-release PR3

API Reference

Last revision date: September 7, 2020

Overview

The current version of InChI Identifier is 1; the current developmental version of the InChI software is 1.06 which is intended to replace the current stable version 1.05 in the near future.

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

The most serious modifications are:
- added support of pseudoelement “Zz” atoms;
- extensions/changes in treatment of polymers;
- security fixes.
Many other minor additions, changes and bugfixes was introduced.

This document provides a brief API reference for InChI Software v. 1.06.

Please note that new features/modifications are marked as "new in v. 1.06" below.

For more details on the related data structures/parameters see inchi_api.h header file in the InChI Software source code).

Classic InChI API

The functions of classic InChI API are considered below. They are mainly the same as in the previous Software version 1.04 and below (see, however, the notes below on more advanced “Ex” (extended functionality) versions and MakeINCHIFromMolfileText() procedure).

Generation of InChI from structure

GetINCHI

```
int INCHI_DECL GetINCHI(inchi_Input *inp, inchi_Output *out)
```

Description

GetINCHI() is the primary function producing InChI. It uses input data in its own inchi_Input format.

GetINCHI produces standard InChI if no InChI creation/stereo modification options are specified. If at least one of the options SUU | SLUUD | RecMet | FixedH | Ket | 15T | SRel | SRac | SUCF is specified, the generated InChI will be non-standard.

Note

It is recommended to use advanced functionality version GetINCHIEx(), see below, instead of this function.

Input

Data structure inchi_Input is created by the user, typically either by reading and parsing Molfile or by conversion from some existing internal molecular representation. Data layout is described in the inchi_api.h header file in the InChI Software source code.

Options supplied to GetINCHI in inchi_Input.szOptions should be preceded by ‘/’ under Windows or ‘-‘ under Linux). Valid options are listed below.

Option	Meaning	Default behavior (standard; if no option supplied)
Structure perception (compatible with standard InChI)		
NEWPSOFF	Both ends of wedge point to stereocenters	Only 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
Stereo interpretation (lead to generation of non-standard InChI)		
SRel	Use relative stereo	Use absolute stereo
SRac	Use racemic stereo	Use absolute stereo
SUCF	Use Chiral Flag in MOL/SD file record: if On – use Absolute stereo, Off – use Relative stereo	Use absolute stereo
ChiralFlagON	Set chiral flag ON	-
ChiralFlagOFF	Set chiral flag OFF	-
InChI creation options (lead to generation of non-standard InChI)		
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)	Stereo labels for “unknown” and “undefined” are the same (‘?’)
FixedH	Include reconnected metals results	Do not include
RecMet	Include Fixed H layer	Do not include

Option	Meaning	Default behavior (standard; if no option supplied)
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
Miscellaneous		
AuxNone	Omit auxiliary information	Include
Wnumber	Set time-out per structure in seconds; W0 means unlimited	The default value is unlimited
Wmnumber	(new in v. 1.06) Set time-out per structure in milliseconds; W0 means unlimited	The default value is unlimited
NoWarnings	(new in v. 1.06) Suppress all warning messages(default: show)	Output warnings as usual
OutputSDF	Output SDfile instead of InChI	-
WarnOnEmptyStructure	Warn and produce empty InChI for empty structure	Just skip empty structure
SaveOpt	Save custom InChI creation options (non-standard InChI)	Do not save custom opts

Output

Data structure inchi_Output is described in the inchi_api.h header file. inchi_Output does not need to be initialized out to zeroes; see FreeNCHI()/FreeSTDINCHI() on how to deallocate it. Strings in inchi_Output are allocated and deallocated by InChI.

Return codes

Code	Value	Meaning
inchi_Ret_OKAY	0	Success; no errors or warnings
inchi_Ret_WARNING	1	Success; warning(s) issued
inchi_Ret_ERROR	2	Error: no InChI has been created
inchi_Ret_FATAL	3	Severe error: no InChI has been created (typically, memory allocation failure)
inchi_Ret_UNKNOWN	4	Unknown program error
inchi_Ret_BUSY	5	Previous call to InChI has not returned yet
inchi_Ret_EOF	-1	No structural data have been provided
inchi_Ret_SKIP	-2	Not used in InChI library

GetINCHIEx

```
int GetINCHIEx( inchi_InputEx *inp, inchi_Output *out )
```

Description

Extended version of GetINCHI() supporting v. 1.05 and further extensions: polymers and Molfile V3000 extended features (partial support).

Note that support of V3000 features is a provisional one: extended data on haptic coordination bonds and stereo collections are read but not used currently (as their inclusion requires significant modification of the InChI identifier itself, not just the Software).

Being able to treat polymer input structures, in other cases this function behaves exactly as the GetINCH() basic API call.

Input

Extended input data structure inchi_InputEx is a superset of inchi_Input of previous versions. The additions are newly included data sub-structures holding information on polymers and V3000 extended features (mostly reflecting a way of description used by Accelrys in Molfiles).

Data structure inchi_InputEx is created by the user, typically either by reading and parsing Molfile or by conversion from some existing internal molecular representation.

Data layout is described in the inchi_api.h header file in the InChI Software source code.

Options supplied to GetINCHIEx in inchi_InputEx.szOptions should be preceded by '/' under Windows or '-' under Linux). Valid options are the same as for GetINCHI plus the additional ones listed below.

Option	Meaning	Default behavior (if no option supplied)
LooseTSACheck	(new in v. 1.06) Relax strictness of tetrahedral stereo ambiguity check for stereo atoms in (large) rings	Use strict criteria (as in v. 1.05 and previous)
Polymers	Experimental support of simple polymers, current mode	Disabled
Polymers105	(new in v. 1.06) Experimental support of simple polymers in legacy v. 1.05 mode	Disabled
NoFrameShift	(new in v. 1.06) Disable polymer CRU frame shift	Attempt CRU frame shift
FoldCRU	(new in v. 1.06)In polymer treatment, try to fold constitutional repeating units which themselves contain repeats	Disabled
NPZz	(new in v. 1.06) Allow non-polymer Zz pseudo element atoms	Disabled
SAtZZ	(new in v. 1.06) Allow stereo at atoms connected to Zz	Disabled
LargeMolecules	Experimental support of molecules up to 32767 atoms	Disabled

Output

The same as for GetINCHI().

Note

Since v. 1.06, this function requires explicitly supplying option “Polymers” to enable support of polymers (or “Polymers105” to request older v. 1.05 compatibility mode; note however that it is planned to be eliminated in future, leaving explicit-pseudo atoms approach the sole mode).

FreeINCHI

```
void INCHI_DECL FreeINCHI(inchi_Output *out)
```

Description

This function should be called to deallocate char* pointers obtained from each GetINCHI call.

Free_inchi_Input

```
void INCHI_DECL Free_inchi_Input( inchi_Input *pInp )
```

Description

To deallocate and write zeroes into the changed members of pInchiInp->pInp call Free_inchi_Input(inchi_Input *pInp).

Get_inchi_Input_FromAuxInfo

```
int INCHI_DECL Get_inchi_Input_FromAuxInfo( char *szInchiAuxInfo,
                                            int bDoNotAddH,
                                            int bDiffUnkUndfStereo,
                                            InchiInpData *pInchiInp )
```

Description

This function creates the input data structure for InChI generation out of the auxiliary information (AuxInfo) string produced by previous InChI generator calls.

This input structure may then be used in conjunction with the GetINCHI API call.

Note the parameter bDiffUnkUndfStereo (if not 0, use different labels for unknown and undefined stereo) appeared in the software v. 1.03.

Input

szInchiAuxInfo

contains ASCIIZ string of InChI output for a single structure or only the AuxInfo line

bDoNotAddH

if 0 then InChI will be allowed to add implicit H

bDiffUnkUndfStereo

if not 0, use different labels for unknown and undefined stereo

pInchiInp

should have a valid pointer pInchiInp->pInp to an empty (all members = 0) inchi_Input structure

Output

The following members of pInp may be filled during the call: atom, num_atoms, stereo0D, num_stereo0D

Return codes

Same as for GetINCHI.

GetStdINCHI

```
int INCHI_DECL GetStdINCHI(inchi_Input *inp, inchi_Output *out)
```

Description

This is a “standard” counterpart of GetINCHI() which may produce only the standard InChI.

Input

The same as for GetINCHI except that perception/creation options supplied in inchi_Input.szOptions may be only:

NEWPSOFF DoNotAddH SNon

Other possible options are:

AuxNone

Wnumber

OutputSDF

WarnOnEmptyStructure

Output

The same as for GetINCHI except for that only standard InChI is produced.

Return codes

The same as for GetINCHI.

FreeStdINCHI

```
void INCHI_DECL FreeStdINCHI(inchi_Output *out)
```

Description

This is a “standard” counterpart of FreeINCHI which should be called to deallocate char* pointers obtained from each GetStdINCHI call.

Free_std_inchi_Input

```
void INCHI_DECL Free_std_inchi_Input( inchi_Input *pInp )
```

Description

This is a “standard” counterpart of Free_inchi_Input

Get_std_inchi_Input_FromAuxInfo

```
int INCHI_DECL Get_std_inchi_Input_FromAuxInfo(char *szInchiAuxInfo,
                                               int bDoNotAddH,
                                               InchiInpData *pInchiInp )
```

Description

This is a “standard” counterpart of Get_std_inchi_Input_FromAuxInfo.

Generation of InChI from structure, step-by-step way

Note that InChI API "modularized" step-by-step version, set of functions using `INCHIGEN` object, is now frozen.
These procedures still work and are retained for compatibility reasons, but will not receive further development.
Presence of new features, starting from v. 1.06, is not guaranteed (though may occasionally appear).

The main purpose of procedures presented below is to modularize the process of InChI generation by separating normalization, canonicalization, and serialization stages. Using these API functions allows, in particular, checking intermediate normalization results before performing further steps and getting diagnostic messages from each stage independently.

The functions use exactly the same inchi_Input and inchi_Output data structures as "classic" InChI API functions do.

However, a new data structure, INCHIGEN_DATA, has been added to expose intermediate results (see inchi_api.h header file).

A typical process of InChI generation with this API calls is as follows.

1) Get handle of a new InChI generator object:
HGen = INCHIGEN_Create();

2) read a molecular structure and use it to initialize the generator:
result = INCHIGEN_Setup(HGen, pGenData, pInp);

3) normalize the structure:
result = INCHIGEN_DoNormalization(HGen, pGenData);
optionally, look at the results;

4) obtain canonical numberings:
result = INCHIGEN_DoCanonicalization(HGen, pGenData);

5) serialize, i.e. produce InChI string:
retcode=INCHIGEN_DoSerialization(HGen,GenData, pResults);

6) reset the InChI generator
INCHIGEN_Reset(HGen, pGenData, pResults);
and go to step 2 to read next structure, or

7) Finally destroy the generator object and free standard InChI library memories:
INCHIGEN_Destroy(HGen);

Note that there are also "standard" counterparts of general-purpose functions; these "standard" API calls described below are retained for compatibility and convenience reasons.

Note that since InChI Software v. 1.06 the step-by-step-creation API *does not* support polymer and pseudo atom extensions.

INCHIGEN_Create

```
INCHIGEN_HANDLE INCHI_DECL INCHIGEN_Create(void)
```

Description

InChI Generator: create generator.

Once the generator is created, it may be used repeatedly for processing the new structures. Before repetitive use, the pair of calls INCHIGEN_Reset / INCHIGEN_Setup should occur.

Returns

The handle of InChI generator object or NULL on failure.

Note: the handle is used just to refer to the internal InChI library object, whose structure is invisible to the user (unless the user chooses to browse the InChI source code). This internal object is initialized and modified through the subsequent calls to INCHIGEN API functions.

INCHIGEN_Setup

```
int INCHI_DECL INCHIGEN_Setup(INCHIGEN_HANDLE HGen,
                              INCHIGEN_DATA * pGenData,
                              inchi_Input * pInp)
```

Description

InChI Generator: initialization stage (storing a specific structure in the generator object).

Note: INCHIGEN_DATA object contains intermediate data visible to the user, in particular, the string accumulating diagnostic messages from all the steps.

Input

INCHIGEN_HANDLE HGen is one obtained through INCHIGEN_Create call.

INCHIGEN_DATA * pGenData is created by the caller. It need not to be initialized.

Data structure inchi_Input * pInp is the same as for GetINCHI.

Return codes

The same as for GetINCHI.

INCHIGEN_DoNormalization

```
int INCHI_DECL INCHIGEN_DoNormalization(INCHIGEN_HANDLE HGen, INCHIGEN_DATA *
pGenData)
```

Description

InChI Generator: perform structure normalization.

Should be called after INCHIGEN_Setup.

Note: INCHIGEN_DATA object explicitly exposes the intermediate normalization data, see inchi_api.h.

Input

INCHIGEN_HANDLE HGen and INCHIGEN_DATA *pGenData as they are after calling INCHIGEN_Setup.

Return codes

The same as for GetINCHI.

INCHIGEN_DoCanonicalization

```
int INCHI_DECL INCHIGEN_DoCanonicalization( INCHIGEN_HANDLE HGen,
                                            INCHIGEN_DATA * pGenData)
```

Description

InChI Generator: perform structure canonicalization.

Should be called after INCHIGEN_DoNormalization.

Input

INCHIGEN_HANDLE HGen and INCHIGEN_DATA *pGenData as they are after calling INCHIGEN_DoNormalization.

Return codes

The same as for GetINCHI.

INCHIGEN_DoSerialization

```
int INCHI_DECL INCHIGEN_DoSerialization(INCHIGEN_HANDLE HGen,
                                        INCHIGEN_DATA * pGenData,
                                        inchi_Output * pResults)
```

Description

InChI Generator: perform InChI serialization.

Should be called after INCHIGEN_DoCanonicalization.

Input

INCHIGEN_HANDLE HGen and INCHIGEN_DATA *pGenData as they are after calling INCHIGEN_DoCanonicalization.

Return codes

The same as for GetINCHI.

INCHIGEN_Reset

```
void INCHI_DECL INCHIGEN_Reset(INCHIGEN_HANDLE HGen,
                               INCHIGEN_DATA * pGenData,
                               inchi_Output * pResults)
```

Description

InChI Generator: reset (use before calling INCHIGEN_Setup(...) to start processing the next structure and before calling INCHIGEN_Destroy(...))

Input

INCHIGEN_HANDLE HGen and INCHIGEN_DATA *pGenData as they are after calling INCHIGEN_DoSerialization.

Return codes

The same as for GetINCHI.

**
**

INCHIGEN_Destroy

```
void INCHI_DECL INCHIGEN_Destroy(INCHIGEN_HANDLE HGen)
```

Description

Destroys the generator object and frees associated InChI library memories.

Important: make sure INCHIGEN_Reset(...) is called before calling INCHIGEN_Destroy(...).

Input

The handle of InChI generator object.

STDINCHIGEN_Create

```
INCHIGEN_HANDLE INCHI_DECL STDINCHIGEN_Create(void)
```

Description

Standard InChI Generator: create generator.

This is a “standard” counterpart of INCHIGEN_Create.

Returns

The handle of standard InChI generator object or NULL on failure. Note: the handle serves to access the internal object, whose structure is invisible to the user (unless the user chooses to browse the InChI library source code which is open).

STDINCHIGEN_Setup

```
int INCHI_DECL STDINCHIGEN_Setup(INCHIGEN_HANDLE HGen,
                                 INCHIGEN_DATA * pGenData,
                                 inchi_Input * pInp)
```

Description

Standard InChI Generator: initialization stage (storing a specific structure in the generator object).

This is a “standard” counterpart of INCHIGEN_Setup.

Note: INCHIGEN_DATA object contains intermediate data visible to the user, in particular, the string accumulating diagnostic messages from all the steps.

Input

INCHIGEN_HANDLE HGen is one obtained through INCHIGEN_Create call.

INCHIGEN_DATA * pGenData is created by the caller.

Data structure inchi_Input * pInp is the same as for GetINCHI.

Return codes

The same as for GetStdINCHI.

STDINCHIGEN_DoNormalization

```
int INCHI_DECL STDINCHIGEN_DoNormalization(INCHIGEN_HANDLE HGen,
                                           INCHIGEN_DATA * pGenData)
```

Description

Standard InChI Generator: perform structure normalization.

The entry is the “standard” counterpart of INCHIGEN_DoNormalization.

STDINCHIGEN_DoCanonicalization

```
int INCHI_DECL STDINCHIGEN_DoCanonicalization(INCHIGEN_HANDLE HGen,
                                              INCHIGEN_DATA * pGenData)
```

Description

Standard InChI Generator: perform structure canonicalization.

The entry is the “standard” counterpart of INCHIGEN_DoCanonicalization.

STDINCHIGEN_DoSerialization

```
int INCHI_DECL STDINCHIGEN_DoSerialization(INCHIGEN_HANDLE HGen,
                                           INCHIGEN_DATA * GenData,
                                           inchi_Output * pResults)
```

Description

Standard InChI Generator: perform InChI serialization.

The entry is the “standard” counterpart of INCHIGEN_DoSerialization.

STDINCHIGEN_Reset

```
void INCHI_DECL STDINCHIGEN_Reset(INCHIGEN_HANDLE HGen,
                                  INCHIGEN_DATA * pGenData,
                                  inchi_Output * pResults)
```

Description

Standard InChI Generator: reset (use before calling STDINCHIGEN_Setup(...) to start processing the next structure and before calling STDINCHIGEN_Destroy(...))

The entry is the “standard” counterpart of INCHIGEN_Reset.

STDINCHIGEN_Destroy

```
INCHI_API void INCHI_DECL STDINCHIGEN_Destroy(INCHIGEN_HANDLE HGen)
```

Description

Destroys the standard InChI generator object and frees associated InChI library memories.

This is the “standard” counterpart of INCHIGEN_Destroy.

Important: make sure STDINCHIGEN_Reset(...) is called before calling STDINCHIGEN_Destroy(...).

Generation of InChI directly from Molfile

MakeINCHIFromMolfileText

```
INCHI_API int INCHI_DECL MakeINCHIFromMolfileText(const char *moltext,
                                                  char *options,
                                                  inchi_Output *result )
```

Description

This function creates InChI from Molfile supplied as a null-terminated string.

That is, it automates reading/parsing Molfile, creation of InChI input and generation of InChI string. Notably, it relies on the same Molfile parser as inchi-1 executable thus ensuring that any correct caller will produce the same result as inchi-1.

Input

moltext	Molfile as null-terminated string
options	the same options as for GetINCHIEx()

Output

The same inchi_Output data structure as for GetNCHI.

Note

Since v. 1.06, this function provides full-scale (though experimental) support of polymers. This requires specifying option “Polymers” (or “Polymers105” to request older v. 1.05 compatibility mode) in input parameter options.

Restoring structure from InChI or AuxInfo

GetStructFromINCHI

```
int INCHI_DECL GetStructFromINCHI(inchi_InputINCHI *inpInChI,
                                  inchi_OutputStruct *outStruct)
```

Description

This function creates structure from InChI string.

Option Inchi2Struct is not needed for GetStructFromINCHI.

Input

Data structure inchi_Inputinchi_InputINCHI is created by the user.

For the description, see header file inchi_api.h.

Output

For the description of inchi_OutputStruct, see header file inchi_api.h. Pointers in inchi_OutputStruct are allocated and deallocated by InChI. inchi_OutputStruct does not need to be initialized out to zeroes; see FreeStructFromINCHI() on how to deallocate it.

Return codes

The same as for GetINCHI.

GetStructFromINCHIEx

```
int INCHI_DECL GetStructFromINCHIEx(inchi_InputINCHI *inpInChI,
                                    inchi_OutputStructEx *outStruct)
```

Description

This extended version of GetStructFromINCHI supports v. 1.05 extensions: polymers and Molfile V3000 (partial support).

Input

The same as for GetStructFromINCHI().

Output

The data structure inchi_OutputStructEx. It is a superset of inchi_OutputStruct including additional data-substructures carrying an information on polymers and V3000 features.
Note that restoring structure from InChI for polymers does not provide information on placement of the polymer-enclosing brackets and on textual index (‘n’ or alike), as the related data are not embedded in InChI string.

For more details on inchi_OutputStructEx data structure, please see inchi_api.h header file in the InChI Software source code.

FreeStructFromINCHI

```
void INCHI_DECL FreeStructFromINCHI( inchi_OutputStruct *out )
```

Description

Should be called to deallocate pointers obtained from each GetStructFromINCHI.

GetStructFromStdINCHI

```
int INCHI_DECL GetStructFromStdINCHI(inchi_InputINCHI *inpInChI,
                                     inchi_OutputStruct *outStruct)
```

Description

This is the "standard" counterpart of GetStructFromINCHI.

Input

The same as for GetStructFromINCHI.

Output

The same as for GetStructFromINCHI.

Return codes

The same as for GetStructFromINCHI.

FreeStructFromStdINCHI

```
void INCHI_DECL FreeStructFromStdINCHI(inchi_OutputStruct *out)
```

Description

Should be called to deallocate pointers obtained from each GetStructFromINCHI.

InChIKey

GetINCHIKeyFromINCHI

```
int INCHI_DECL GetINCHIKeyFromINCHI(const char* szINCHISource,
                                    const int xtra1,
                                    const int xtra2,
                                    char* szINCHIKey,
                                    char* szXtra1,
                                    char* szXtra2)
```

Description

Calculate InChIKey from InChI string.

Input

szINCHISource – source null-terminated InChI string.

xtra1 =1 calculate hash extension (up to 256 bits; 1st block)

xtra2 =1 calculate hash extension (up to 256 bits; 2nd block)

Output

szINCHIKey - InChIKey string, null-terminated. The user-supplied buffer szINCHIKey should be at least 28 bytes long.

szXtra1- hash extension (up to 256 bits; 1st block) string. Caller should allocate space for 64 characters + trailing NULL.

szXtra2 - hash extension (up to 256 bits; 2nd block) string. Caller should allocate space for 64 characters + trailing NULL.

Return codes

Code	Value	Meaning
INCHIKEY_OK	0	Success; no errors or warnings
INCHIKEY_UNKNOWN_ERROR	1	Unknown program error
INCHIKEY_EMPTY_INPUT	2	Source string is empty
INCHIKEY_INVALID_INCHI_PREFIX	3	Invalid InChI prefix or invalid version (not 1)
INCHIKEY_NOT_ENOUGH_MEMORY	4	Not enough memory
INCHIKEY_INVALID_INCHI	20	Source InChI has invalid layout
INCHIKEY_INVALID_STD_INCHI	21	Source standard InChI has invalid layout

CheckINCHIKey

```
int INCHI_DECL CheckINCHIKey(const char *szINCHIKey)
```

Description

Check if the string represents valid InChIKey.

Input

szINCHIKey - source InChIKey string

Return codes

Code	Value	Meaning
INCHIKEY_VALID_STANDARD	0	InChIKey is valid and standard
INCHIKEY_VALID_NON_STANDARD	-1	InChIKey is valid and non-standard
INCHIKEY_INVALID_LENGTH	1	InChIKey has invalid length
INCHIKEY_INVALID_LAYOUT	2	InChIKey has invalid layout
INCHIKEY_INVALID_VERSION	3	InChIKey has invalid version number (not equal to 1)

GetStdINCHIKeyFromStdINCHI

```
int INCHI_DECL GetStdINCHIKeyFromStdINCHI(const char* szINCHISource,
                                          char* szINCHIKey)
```

Description

Calculate standard InChIKey from standard InChI string.

"Standard" counterpart of GetINCHIKeyFromINCHI.

For compatibility with v. 1.02-standard, no extra hash calculation is allowed. To calculate extra hash(es), use GetINCHIKeyFromINCHI with stdInChI as input.

Input

szINCHISource – source null-terminated InChI string.

Output

szINCHIKey - InChIKey string, null-terminated. The user-supplied buffer szINCHIKey should be at least 28 bytes long.

Return codes

The same as for GetINCHIKeyFromINCHI.

Test and utlity procedures

GetINCHIfromINCHI

```
int INCHI_DECL GetINCHIfromINCHI(inchi_InputINCHI *inpInChI,
                                 inchi_Output *out)
```

Description

GetINCHIfromINCHI does the same as the -InChI2InChI option: converts InChI into InChI for validation purposes. It may also be used to filter out specific layers. For instance, SNon would remove the stereochemical layer. Omitting FixedH and/or RecMet would remove Fixed-H or Reconnected layers. Option InChI2InChI is not needed.

Notes: options are supplied in inpInChI[AM1] .szOptions. Options should be preceded by '/' under Windows or '-' under Linux; there is no explicit tool to conversion from/to standard InChI

Input

inchi_InputINCHI is created by the user.

Output

Strings in inchi_Output are allocated and deallocated by InChI. inchi_Output does not need to be initialized out to zeroes; see FreeINCHI() on how to deallocate it.

Return codes

Same as for GetINCHI.

CheckINCHI

```
int INCHI_DECL CheckINCHI(const char *szINCHI, const int strict)
```

Description

Check if the string represents valid InChI/standard InChI.

Input

Input:

szINCHI source InChI

strict if 0, just briefly check for proper layout (prefix, version, etc.).

The result may not be strict.

If not 0, try to perform InChI2InChI conversion; returns success if a resulting InChI string exactly matches source. Be cautious: the result may be too strict, i.e. a 'false alarm', due to imperfection of conversion.

Return codes

Code	Value	Meaning
INCHI_VALID_STANDARD	0	InChI is valid and standard
INCHI_VALID_NON_STANDARD	-1	InChI is valid and non-standard
INCHI_INVALID_PREFIX	1	InChI has invalid prefix
INCHI_INVALID_VERSION	2	InChI has invalid version number (not equal to 1)
INCHI_INVALID_LAYOUT	3	InChI has invalid layout
INCHI_FAIL_I2I	4	Checking InChI through InChI2InChI either failed or produced a result which does not match the source InChI string

GetStringLength

```
int INCHI_DECL GetStringLength( char *p )
```

Description

Returns string length.

InChI Extensible API – IXA

The InChI Extensible API provides an alternative access to all the functionality in the original API. The primary purpose of the IXA is to ensure complete separation of the interface to the underlying InChI generation code from the implementation of that code. This will permit changes to be made to the implementation, as well as development and extension of the InChI code to handle new types of structure, without affecting the interface, or user code which is dependent on that interface.

The IXA provides both low-level and high-level means of specifying molecules. The low level approach involves specifying the individual atoms and bonds and their properties, in a series of calls to separate functions. The high level approach specifies a complete molecule in a single call which reads, for example, an MDL Molfile, or an InChI.

IXA is defined in the ISO standard C language and is based on the use of several different Object types, which are accessed by means of “Handles”. Each function in the IXA operates on one or more of these Objects.

The Objects defined in the IXA are as follows:

- Status Objects, containing error and warning messages
- Molecule Objects, containing representations of molecules or other chemical entities
- InChI Builder Objects, used to construct InChI strings
- InChIKey Builder Objects, used to construct InChIKeys

The Handle for each of variety of Object has its own C type, which ensures that the Handles for different varieties of Object cannot be confused or interchanged. Functions are provided for the creation and destruction of Objects, as well as for modifying and manipulating them in various ways, and these functions are responsible for all allocation and freeing of memory used by the Objects.

The details of Objects and related functions are as follow.

Status Objects

IXA Status Objects are used to accumulate error and warning messages generated by the functions in the IXA. Most functions in the IXA require the Handle for an IXA Status Object to be passed as a parameter; any error or warning messages generated by the function are then stored in the IXA Status Object.

IXA Status Objects can be interrogated to discover how many messages they have accumulated, the severity of those messages (error or warning), and of course, to obtain the text of each individual message. A function is also provided to clear all messages in the IXA Status Object.

Generally, a user program will start by creating an IXA Status Object, and will then pass its Handle to all subsequent IXA function calls, checking for messages after each call or group of calls to ensure that they have been successful. As a general principle, the value returned by an IXA function should not be used to determine whether or not an error has occurred – the documentation for each function generally notes the value that is returned on error, though in many cases this value can also be returned when no error has occurred.

Types and Constants

IXA Status Object Handles have type IXA_STATUS_HANDLE.

The severity of a status message is given in variables of type IXA_STATUS, which has

- IXA_STATUS_SUCCESS: An operation was successful, and generated no messages.
- IXA_STATUS_WARNING: An operation was successful, but generated a warning message.
- IXA_STATUS_ERROR: An operation failed with an error message.

Some functions take Boolean (TRUE/FALSE) parameters, or return Boolean values expressed using the special type IXA_BOOL, which has the following enumerated constants:

- IXA_FALSE
- IXA_TRUE.

Functions

IXA_STATUS_Create

```
IXA_STATUS_HANDLE IXA_STATUS_Create ( )
```

Description

Creates a new IXA Status Object and returns its Handle.

Input

None

Output

Handle for the newly-created IXA Status Object.

IXA_STATUS_Clear

```
void IXA_STATUS_Clear (IXA_STATUS_HANDLE hStatus)
```

Description

Clears all messages held by an IXA Status Object.

Input

hStatus: Handle for the IXA Status Object to be cleared.

IXA_STATUS_Destroy

```
void IXA_STATUS_Destroy (IXA_STATUS_HANDLE hStatus)
```

Description

Destroys an IXA Status Object, releasing all memory that it uses.

Input

hStatus: Handle for the IXA Status Object to be destroyed.

IXA_STATUS_HasError

```
IXA_BOOL IXA_STATUS_HasError (IXA_STATUS_HANDLE hStatus)
```

Description

Returns IXA_TRUE if an IXA Status Object holds a message with severity IXA_STATUS_ERROR.

Input

hStatus: Handle for the IXA Status Object to be examined.

Output

IXA_TRUE if the IXA Status Object holds a message with severity IXA_STATUS_ERROR;

IXA_FALSE if it does not, or if hStatus is invalid.

IXA_STATUS_HasWarning

```
IXA_BOOL IXA_STATUS_HasWarning (IXA_STATUS_HANDLE hStatus)
```

Description

Returns IXA_TRUE if an IXA Status Object holds a message with severity IXA_STATUS_WARNING.

Input

hStatus: Handle for the IXA Status Object to be examined.

Output

IXA_TRUE if the IXA Status Object holds a message with severity IXA_STATUS_WARNING; IXA_FALSE if it does not, or if hStatus is invalid.

IXA_STATUS_GetCount

```
int IXA_STATUS_GetCount (IXA_STATUS_HANDLE hStatus)
```

Description

Returns the total number of status messages held by an IXA Status Object.

Input

hStatus: Handle for the IXA Status Object to be examined.

Output

The total number of status messages held by the IXA Status Object, or zero if hStatus is invalid.

IXA_STATUS_GetSeverity

```
IXA_STATUS IXA_STATUS_GetSeverity (IXA_STATUS_HANDLE hStatus,
                                   int vIndex)
```

Description

Returns the severity of a status message held by an IXA Status Object.

Input

hStatus: Handle for the IXA Status Object to be examined.

vIndex Index number (from zero) of the status message to be examined.

Output

Severity of the specified status message in the IXA Status Object. IXA_STATUS_ERROR if

hStatus is invalid or vIndex is out of range.

IXA_STATUS_GetMessage

```
const char* IXA_STATUS_GetMessage (IXA_STATUS_HANDLE hStatus,
                                   int vIndex)
```

Description

Returns the text of a status message held by an IXA Status Object.

Input

hStatus: Handle for the IXA Status Object to be examined.

vIndex: Index number (from zero) of the status message to be returned.

Output

Text of the specified status message in the IXA Status Object, or NULL if hStatus is invalid or vIndex is out of range. The returned string is null-terminated and is owned by the IXA Status Object, and must be copied by the user if it is to be retained.

Molecule Objects

IXA Molecule Objects are used to represent molecules, with their constituent atoms, bonds and stereo descriptors.

IXA Molecule Objects are initially created empty, and can be populated either in single function calls (for example by reading a Molfile or an InCHI), or by successively adding individual atoms, bonds and stereodescriptors, and specifying their properties, in separate function calls. Functions are also provided to return information about the atoms, bonds and stereodescriptors in an IXA Molecule Object.

Within an IXA Molecule Object, each individual atom, bond or stereodescriptor has a unique Identifier, which like the Handles for the main IXA Objects, have their own C types.

Stereochemistry

Two mechanisms are provided for the representation of stereochemistry in IXA Molecule Objects.

The first of these allows specification of special stereochemical properties for individual bonds within an IXA Molecule Object – “up” and “down” wedges etc. on single bonds, and an indication as to whether or not the X/Y coordinates of atoms around double bonds should be used to determine their configuration. This mechanism is dependent on appropriate coordinates being specified for the atoms, and even then it is possible for ambiguous or self-contradictory configurations to be specified using it; it is meaningless if 2D coordinates are not available.

The second mechanism uses a separate stereodescriptor, with its own IXA Identifier, for each stereocentre. The stereodescriptor specifies the topology involved, identifies the central atom or bond, lists the vertices that surround it and specifies the “parity” for the stereocentre. This type of stereodescriptor is the only way of specifying stereochemistry within IXA Molecule Objects if coordinates are not available, and is used for IXA Molecule Objects populated from InChIs (which do not record coordinates).

Types and Constants

IXA Molecule Object Handles have type IXA_MOL_HANDLE.

IXA Atom Identifiers have type IXA_ATOMID and there are two special constants of this type. IXA_ATOMID_INVALID is the Identifier for an invalid atom within an IXA Molecule Object, and is the value returned by some functions when a error occurs. IXA_ATOMID_IMPLICIT_H is the Identifier for an implicit hydrogen atom attached to another atom, and is the value used to specify implicit hydrogen atoms when specifying stereocentres.

Atom radical states are specified by constants of type IXA_ATOM_RADICAL with possible values:

- IXA_ATOM_RADICAL_NONE: The atom is not a radical.

- IXA_ATOM_RADICAL_SINGLET: The atom is a singlet radical.

- IXA_ATOM_RADICAL_DOUBLET: The atom is a doublet radical.

- IXA_ATOM_RADICAL_TRIPLET: The atom is a triplet radical.

IXA Bond Identifiers have type IXA_BONDID; IXA_BONDID_INVALID is a special constant of type IXA_BONDID, and is the Identifier for an invalid bond within an IXA Molecule Object; it is the value returned by some functions when an error occurs.

Bond types within IXA Molecule Objects have type IXA_BOND_TYPE with possible values: • IXA_BOND_TYPE_SINGLE: The bond is a single bond.

- IXA_BOND_TYPE_DOUBLE: The bond is a double bond.

- IXA_BOND_TYPE_TRIPLE: The bond is a triple bond.

- IXA_BOND_TYPE_AROMATIC: The bond is an “aromatic” bond.

As part of the InChI generation process, aromatic bonds are replaced by patterns of single and double bonds; where this cannot be done, appropriate error or warning messages may be issued. Where single-bond stereochemistry is indicated by “wedge bonds”, the wedge direction is shown by a bond property of type IXA_BOND_WEDGE with possible values:

- IXA_BOND_WEDGE_NONE: The bond has no wedge property; this is the default value where no stereochemistry is involved.

- IXA_BOND_WEDGE_UP: The wedge points “up” from the reference atom.

- IXA_BOND_WEDGE_DOWN: The wedge points “down” from the reference atom.

- IXA_BOND_WEDGE_EITHER: The wedge can point either “up” or “down” from the reference atom.

The stereochemical configuration for double bonds is specified by a bond property of type IXA_DBLBOND_CONFIG with possible values:

- IXA_DBLBOND_CONFIG_PERCEIVE: The configuration (if any) should be perceived from the X and Y coordinates of the atoms joined by the bond and their neighbours.

- IXA_DBLBOND_CONFIG_EITHER: The bond can be in either configuration.

IXA Stereodescriptor Identifiers have type IXA_STEREOID; IXA_STEREOID_INVALID is a special constant of type IXA_STEREOID and is the Identifier for an invalid stereodescriptor within an IXA Molecule Object; it is the value returned by some functions when an error occurs.

The topology described by an IXA Stereodescriptor is specified by constants of type IXA_STEREO_TOPOLOGY with possible values:

- IXA_STEREO_TOPOLOGY_TETRAHEDRON: The atoms around a central atom are arranged in a tetrahedron – e.g. sp3 carbon.

- IXA_STEREO_TOPOLOGY_RECTANGLE: The atoms around a central bond are arranged in a rectangle – e.g. olefins, and cumulenes.

- IXA_STEREO_TOPOLOGY_ANTIRECTANGLE: The atoms around a central atom are arranged in an anti-rectangle – e.g. allenes.

- IXA_STEREO_TOPOLOGY_INVALID: Used as a return value in case of errors.

The stereo parity described by an IXA Stereodescriptor is specified by constants of type IXA_STEREO_PARITY with possible values:

- IXA_STEREO_PARITY_NONE: No parity value is defined for the stereocentre.

- IXA_STEREO_PARITY_ODD: The stereocentre has odd parity.

- IXA_STEREO_PARITY_EVEN: The stereocentre has even parity.

- IXA_STEREO_PARITY_UNKNOWN: The parity of the stereocentre is unknown.

IXA polymer unit Identifiers have type IXA_POLYMERUNITID; IXA_POLYMERUNITID_INVALID is a special constant of type IXA_POLYMERUNITID and is the Identifier for an invalid monomeric unit within an IXA Molecule Object; it is the value returned by some functions when an error occurs.

Functions to Create, Clear and Destroy Molecule Objects

IXA_MOL_Create

```
IXA_MOL_HANDLE IXA_MOL_Create (IXA_STATUS_HANDLE hStatus)
```

Description

Creates a new empty IXA Molecule Object and returns its Handle.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

Output

Handle for the newly-created IXA Molecule Object.

IXA_MOL_Clear

```
void IXA_MOL_Clear (IXA_STATUS_HANDLE hStatus,
                    IXA_MOL_HANDLE hMolecule)
```

Description

Clears all data in an IXA Molecule Object, returning it to an empty state as when newly created.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be cleared.

IXA_MOL_Destroy

```
void IXA_MOL_Destroy(IXA_STATUS_HANDLE hStatus,
                     IXA_MOL_HANDLE hMolecule)
```

Description

Destroys an IXA Molecule Object, releasing all memory that it uses.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be destroyed.

Functions Operating on Complete Molecules

These functions operate on IXA Molecule Objects at “high level”, and do not require access to individual atoms, bonds and stereodescriptors.

IXA_MOL_ReadMolfile

```
void IXA_MOL_ReadMolfile(IXA_STATUS_HANDLE hStatus,
                         IXA_MOL_HANDLE hMolecule,
                         const char* pMolfile)
```

Description

Populates an IXA Molecule Object with data from an MDL Molfile representation. Any data previously held in the IXA Molecule Object are over-written.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule Handle for the IXA Molecule Object to be populated.

pMolfile Null-terminated character array containing the text of the Molfile. Reading continues until the syntactic end of the Molfile is reached, or until a null character is reached, whichever occurs first.

IXA_MOL_ReadInChI

```
void IXA_MOL_ReadInChI(IXA_STATUS_HANDLE hStatus,
                       IXA_MOL_HANDLE hMolecule,
                       const char* pInChI)
```

Description

Populates an IXA Molecule Object with data from an InChI string representation. Any data

previously held in the IXA Molecule Object are over-written.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule Handle for the IXA Molecule Object to be populated.

pInChI Null-terminated character array containing the an InChI string. Reading continues until the syntactic end of the InChI is reached, or until a null character is reached, whichever occurs first.

Output

Nothing

IXA_MOL_SetChiral

```
void IXA_MOL_SetChiral(IXA_STATUS_HANDLE hStatus,
                       IXA_MOL_HANDLE hMolecule,
                       IXA_BOOL vChiral)
```

Description

Sets the chiral flag for an IXA Molecule Object. If the non-standard InChI generation option IXA_INCHIBUILDER_STEREOOPTION_SUCF is specified, the chiral flag is used to determine how stereochemistry in the IXA Molecule Object should be interpreted.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vChiral: Value to be used for the chiral flag (IXA_TRUE = molecule is chiral; IXA_FALSE = molecule is not chiral).

Output

Nothing

IXA_MOL_GetChiral

```
IXA_BOOL IXA_MOL_GetChiral(IXA_STATUS_HANDLE hStatus,
                           IXA_MOL_HANDLE hMolecule)
```

Description

Returns the value of the chiral flag for an IXA Molecule Object. If the non-standard InChI generation option IXA_INCHIBUILDER_STEREOOPTION_SUCF is specified, the chiral flag is used to determine how stereochemistry in the IXA Molecule Object should be interpreted.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

Output

Value of chiral flag (IXA_TRUE = molecule is chiral; IXA_FALSE = molecule is not chiral).

Functions to Add and Define Atoms

When an individual atom is created in an IXA Molecule Object, it has a set of default properties (carbon with IXA_ATOM_NATURAL_MASS, radical state IXA_ATOM_RADICAL_NONE, zero for all numerical properties other than atomic number, and no bonds to other atoms) which can then be modified if required.

IXA_MOL_CreateAtom

```
IXA_ATOMID IXA_MOL_CreateAtom(IXA_STATUS_HANDLE hStatus,
                              IXA_MOL_HANDLE hMolecule)
```

Description

Adds one atom to an IXA Molecule Object, and returns its IXA Atom Identifier. The atom is set to be a carbon atom with mass IXA_ATOM_NATURAL_MASS, and no bonds to other atoms. Its radical state is set to IXA_ATOM_RADICAL_NONE, and all its numerical properties (other than atomic number) are set to zero.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

Output

IXA Atom Identifier for the newly-created atom, or IXA_ATOMID_INVALID on error.

IXA_MOL_SetAtomElement

```
void IXA_MOL_SetAtomElement(IXA_STATUS_HANDLE hStatus,
                            IXA_MOL_HANDLE hMolecule,
                            IXA_ATOMID vAtom,
                            const char* pElement)
```

Description

Sets the element type for an atom in an IXA Molecule Object. The element type can also be set by function IXA_MOL_SetAtomAtomicNumber.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vAtom: IXA Atom Identifier for the atom to be modified.

pElement: Null-terminated character string containing the IUPAC element symbol to be used for the specified atom. All IUPAC-approved two-letter symbols up to the element 118.

IXA_MOL_SetAtomAtomicNumber

```
void IXA_MOL_SetAtomAtomicNumber(IXA_STATUS_HANDLE hStatus,
                                 IXA_MOL_HANDLE hMolecule,
                                 IXA_ATOMID vAtom,
                                 int vAtomicNumber)
```

Description

Sets the atomic number for an atom in an IXA Molecule Object. The atomic number can also be set by function IXA_MOL_SetAtomElement.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vAtom: IXA Atom Identifier for the atom to be modified.

vAtomicNumber: The atomic number to be used for the specified atom. Valid values are in the range 1-118 inclusive.

IXA_MOL_SetAtomMass

```
void IXA_MOL_SetAtomMass(IXA_STATUS_HANDLE hStatus,
                         IXA_MOL_HANDLE hMolecule,
                         IXA_ATOMID vAtom,
                         int vMassNumber)
```

Description

Sets the mass number for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vAtom: IXA Atom Identifier for the atom to be modified.

vMassNumber: The mass number to be used for the specified atom. The constant IXA_ATOM_NATURAL_MASS may be used to specify the naturally-abundant mixture of masses, which is the default.

IXA_MOL_SetAtomCharge

```
void IXA_MOL_SetAtomCharge(IXA_STATUS_HANDLE hStatus,
                           IXA_MOL_HANDLE hMolecule,
                           IXA_ATOMID vAtom,
                           int vCharge)
```

Description

Sets the formal charge on an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule Handle for the IXA Molecule Object to be modified.

vAtom IXA Atom Identifier for the atom to be modified.

vCharge The charge to be used for the specified atom. No constraints are imposed on the

permitted range of values.

IXA_MOL_SetAtomRadical

```
void IXA_MOL_SetAtomRadical(IXA_STATUS_HANDLE hStatus,
                            IXA_MOL_HANDLE hMolecule,
                            IXA_ATOMID vAtom,
                            IXA_ATOM_RADICAL vRadical)
```

Description

Sets the radical state for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vAtom: IXA Atom Identifier for the atom to be modified.

vRadical: The radical state constant to be used for the specified atom.

IXA_MOL_SetAtomHydrogens

```
void IXA_MOL_SetAtomHydrogens(IXA_STATUS_HANDLE hStatus,
                              IXA_MOL_HANDLE hMolecule,
                              IXA_ATOMID vAtom,
                              int vHydrogenMassNumber,
                              int vHydrogenCount)
```

Description

Sets the number and mass of hydrogen atoms attached to an atom in an IXA Molecule Object.

Multiple calls to this function are permitted to set counts for different hydrogen isotopes attached to the same atom.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vAtom: IXA Atom Identifier for the atom to be modified.

vHydrogenMassNumber: The mass number of the attached hydrogen atoms (in the range 1-3).

vHydrogenCount: The number of hydrogen atoms of the specified mass which are

to be attached to the specified atom.

IXA_MOL_SetAtomX

```
void IXA_MOL_SetAtomX(IXA_STATUS_HANDLE hStatus,
                      IXA_MOL_HANDLE hMolecule,
                      IXA_ATOMID vAtom,
                      double vX)
```

Description

Sets the *x*-coordinate for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vAtom: IXA Atom Identifier for the atom to be modified.

vX: *x*-coordinate to be set.

IXA_MOL_SetAtomY

```
void IXA_MOL_SetAtomY(IXA_STATUS_HANDLE hStatus,
                      IXA_MOL_HANDLE hMolecule,
                      IXA_ATOMID vAtom,
                      double vY)
```

Description

Sets the *y*-coordinate for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vAtom: IXA Atom Identifier for the atom to be modified.

vY: *y*-coordinate to be set.

IXA_MOL_SetAtomZ

```
void IXA_MOL_SetAtomZ(IXA_STATUS_HANDLE hStatus,
                      IXA_MOL_HANDLE hMolecule,
                      IXA_ATOMID vAtom,
                      double vZ)
```

Description

Sets the *z*-coordinate for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vAtom: IXA Atom Identifier for the atom to be modified.

vZ: *z*-coordinate to be set.

Functions to Add and Define Bonds

When an individual bond is created in IXA Molecule Objects, it has a set of default properties (IXA_BOND_TYPE_SINGLE with wedge direction IXA_BOND_WEDGE_NONE with respect to both its atoms) which can then be modified if required.

IXA_MOL_CreateBond

```
IXA_BONDID IXA_MOL_CreateBond (IXA_STATUS_HANDLE hStatus,
                               IXA_MOL_HANDLE hMolecule,
                               IXA_ATOMID vAtom1,
                               IXA_ATOMID vAtom2)
```

Description

Creates a new bond between the specified atoms in an IXA Molecule Object, and returns its IXA Bond Identifier. By default, the bond created has bond type IXA_BOND_TYPE_SINGLE and its wedge direction is IXA_BOND_WEDGE_NONE. In the event that it is changed to a double bond, its double bond configuration is IXA_DBLBOND_CONFIG_PERCEIVE.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vAtom1: IXA Atom Identifier for the atom at one end of the new bond.

vAtom2: IXA Atom Identifier for the atom at the other end of the new bond.

Output

The IXA Bond Identifier for the new bond, or IXA_BONDID_INVALID on error.

IXA_MOL_SetBondType

```
void IXA_MOL_SetBondType(IXA_STATUS_HANDLE hStatus,
                         IXA_MOL_HANDLE hMolecule,
                         IXA_BONDID vBond,
                         IXA_BOND_TYPE vType)
```

Description

Sets the bond type for a bond in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vBond: IXA Bond Identifier for the bond to be modified.

vType: The bond type to be used for the specified bond.

IXA_MOL_SetBondWedge

```
void IXA_MOL_SetBondWedge(IXA_STATUS_HANDLE hStatus,
                          IXA_MOL_HANDLE hMolecule,
                          IXA_BONDID vBond,
                          IXA_ATOMID vRefAtom,
                          IXA_BOND_WEDGE vDirection)
```

Description

Sets the wedge direction for a single bond in an IXA Molecule Object with respect to a specified atom. This property is only relevant for IXA_BOND_TYPE_SINGLE bonds. Note that wedge direction is associated with the reference atom only; setting a wedge direction for a bond with respect to one atom does not set a wedge direction for the same bond with respect to its other atom.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vBond: IXA Bond Identifier for the bond to be modified.

vRefAtom: IXA Atom Identifier for the reference atom, at one end of the specified bond.

vDirection: The wedge direction to be used for the specified bond with respect to the specified atom.

IXA_MOL_SetDblBondConfig

```
void IXA_MOL_SetDblBondConfig(IXA_STATUS_HANDLE hStatus,
                              IXA_MOL_HANDLE hMolecule,
                              IXA_BONDID vBond,
                              IXA_DBLBOND_CONFIG vConfig)
```

Description

Sets the stereo configuration for a double bond in an IXA Molecule Object. This property is only relevant for IXA_BOND_TYPE_DOUBLE bonds.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vBond: IXA Bond Identifier for the bond to be modified.

vConfig: The bond configuration to be used for the specified bond.

Functions to Add and Define Stereodescriptors

Each individual stereodescriptor in an IXA Molecule Object describes the configuration at a single stereocentre. This is done by specifying the geometry of the stereocentre, the central atom or bond, and the vertices which surround it. Separate creation functions are provided for each geometry, as the number of vertices involved may vary between geometries. Where one of the vertices to be specified is an “implicit hydrogen” with no IXA Atom Identifier of its own, the constant IXA_ATOMID_IMPLICIT_H should be used.

IXA_MOL_CreateStereoTetrahedron

```
IXA_STEREOID IXA_MOL_CreateStereoTetrahedron(IXA_STATUS_HANDLE hStatus,
                                             IXA_MOL_HANDLE hMolecule,
                                             IXA_ATOMID vCentralAtom,
                                             IXA_ATOMID vVertex1,
                                             IXA_ATOMID vVertex2,
                                             IXA_ATOMID vVertex3,
                                             IXA_ATOMID vVertex4)
```

Description

Creates a new stereodescriptor for a tetrahedral stereocentre in an IXA Molecule Object, and returns its Identifier. The parity for the new stereodescriptor is set to IXA_MOL_STEREOPARITY_NONE on creation and can be modified by function IXA_MOL_SetStereoParity.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vCentralAtom: IXA Atom Identifier for the central atom of the stereocentre.

vVertex1: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the first vertex attached to the stereocentre.

vVertex2: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the second vertex attached to the stereocentre.

vVertex3: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the third vertex attached to the stereocentre.

vVertex4: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the fourth vertex attached to the stereocentre.

Output

IXA Stereodescriptor Identifier for the new stereocentre.

IXA_MOL_CreateStereoRectangle

```
IXA_STEREOID IXA_MOL_CreateStereoRectangle(IXA_STATUS_HANDLE hStatus,
                                           IXA_MOL_HANDLE hMolecule,
                                           IXA_BONDID vCentralBond,
                                           IXA_ATOMID vVertex1,
                                           IXA_ATOMID vVertex2,
                                           IXA_ATOMID vVertex3,
                                           IXA_ATOMID vVertex4)
```

Description

Creates a new stereodescriptor for a rectangular stereocentre (e.g. olefin or cumulene) in an IXA Molecule Object, and returns its Identifier. The parity for the new stereodescriptor is set to IXA_MOL_STEREOPARITY_NONE on creation and can be modified by function IXA_MOL_SetStereoParity.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vCentralBond: IXA Bond Identifier for the central bond of the stereocentre.

vVertex1: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the first vertex attached to the stereocentre.

vVertex2: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the second vertex attached to the stereocentre.

vVertex3: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the third vertex attached to the stereocentre.

vVertex4: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the fourth vertex attached to the stereocentre.

Output

IXA Stereodescriptor Identifier for the new stereocentre.

Note:

In the case of olefins, the stereocentre consists of a double bond, which should be specified as vCentralBond. The four atoms that have bonds to the atoms at either end of vCentralBond should be specified as the four vertices (two at each end of the double bond). In the case of cumulenes, the stereocentre consists of three consecutive double bonds; the central one of these should be specified as vCentralBond. The four atoms that have bonds to the atoms at either end of the cumulated system should be specified as the four vertices (two at each end). In neither case should the atoms involved in any of the double bonds be specified as vertices.

IXA_MOL_CreateStereoAntiRectangle

```
IXA_STEREOID IXA_MOL_CreateStereoAntiRectangle(IXA_STATUS_HANDLE hStatus,
                                               IXA_MOL_HANDLE hMolecule,
                                               IXA_ATOMID vCentralAtom,
                                               IXA_ATOMID vVertex1,
                                               IXA_ATOMID vVertex2,
                                               IXA_ATOMID vVertex3,
                                               IXA_ATOMID vVertex4)
```

Description

Creates a new stereodescriptor for an anti-rectangular stereocentre (e.g. allenic) in an IXA Molecule Object, and returns its Identifier. The parity for the new stereodescriptor is set to IXA_MOL_STEREOPARITY_NONE on creation and can be modified by function IXA_MOL_SetStereoParity.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vCentralAtom: IXA Atom Identifier for the central atom of the stereocentre.

vVertex1: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the first vertex attached to the stereocentre.

vVertex2: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the second vertex attached to the stereocentre.

vVertex3: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the third vertex attached to the stereocentre.

vVertex4: IXA Atom Identifier (or IXA_ATOMID_IMPLICIT_H) for the fourth vertex attached to the stereocentre.

Output

IXA Stereodescriptor Identifier for the new stereocentre.

Note:

In allenes, the stereocentre consists of two consecutive double bonds; the atom between them should be specified as vCentralAtom. The four atoms that have bonds to the atoms at either end of the system should be specified as the four vertices (two at each end). The atoms involved in the double bonds themselves should not be specified as vertices.

IXA_MOL_SetStereoParity

```
void IXA_MOL_SetStereoParity(IXA_STATUS_HANDLE hStatus,
                             IXA_MOL_HANDLE hMolecule,
                             IXA_STEREOID vStereo,
                             IXA_STEREO_PARITY vParity)
```

Description

Sets the parity for a stereodescriptor (specified by an IXA Stereodescriptor Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vStereo: IXA Stereodescriptor Identifier for the stereodescriptor to be modified.

vParity: The parity value to be used for the specified stereodescriptor in the specified molecule.

Functions to Add and Define Polymer Units

IXA_MOL_CreatePolymerUnit (new in v. 1.06)

```
IXA_POLYMERUNITID IXA_MOL_CreatePolymerUnit(IXA_STATUS_HANDLE hStatus,
                                            IXA_MOL_HANDLE hMolecule)
```

Description

Creates a new polymer unit in an IXA Molecule Object, and returns its Identifier. The properties of a new unit is set by IXA_MOL_SetPolymerUnit API procedure.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

Output

IXA polymer unit Identifier for the new unit.

IXA_MOL_GetPolymerUnitId (new in v. 1.06)

```
IXA_POLYMERUNITID INCHI_DECL IXA_MOL_GetPolymerUnitId(IXA_STATUS_HANDLE hStatus,
                                                      IXA_MOL_HANDLE hMolecule,
                                                      int vPolymerUnitIndex)
```

IXA_MOL_GetPolymerUnitIndex (new in v. 1.06)

```
int IXA_MOL_GetPolymerUnitIndex(IXA_STATUS_HANDLE hStatus,
                                IXA_MOL_HANDLE  hMolecule,
                                IXA_POLYMERUNITID vPolymerUnit)
```

IXA_MOL_SetPolymerUnit (new in v. 1.06)

```
 void IXA_MOL_SetPolymerUnit(IXA_STATUS_HANDLE hStatus,
                             IXA_MOL_HANDLE hMolecule,
                             IXA_POLYMERUNITID vPunit,
                             int vId,
                             int vType,
                             int vSubtype,
                             int vConn,
                             int vLabel,
                             int vNa,
                             int vNb,
                             double vXbr1[4],
                             double vXbr2[4],
                             char vSmt[80],
                             int *vAlist,
                             int *vBlist)
```

Description

Sets the formal charge on an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be modified.

vPunit: IXA Atom Identifier for the unit to be modified.

vId: 'Sgroup number', see CTFILE description.

vType: type as by MDL format (STY)

vSubtype: subtype as by MDL format (SST)

vConn: connection scheme as by MDL format (SCN)

vLabel: it is what is called 'unique Sgroup identifier' in CTFILE

vNa: number of atoms in the unit

vNb: number of bonds in the unit

vXbr1[4]: bracket ends coordinates (SDI)

vXbr2[4]: bracket ends coordinates (SDI)

vSmt[80]: Sgroup Subscript (SMT)

*vAlist: atom numbers [num_atom1, num_atom2, num_atom3,..] for atom in CRU (SAL)

*vBlist: bonds in unit [num_atom1, num_atom2, num_atom1, num_atom2,..] for all bonds (as made from SBL)

Functions to Navigate Within a Molecule

The functions described in this section return information about which atoms are connected by which bonds in an IXA Molecule Object, and allow navigation within it.

IXA_MOL_GetNumAtoms

```
int IXA_MOL_GetNumAtoms(IXA_STATUS_HANDLE hStatus,
                        IXA_MOL_HANDLE hMolecule)
```

Description

Returns the number of atoms in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

Output

Total number of atoms (not counting implicit hydrogens) in the IXA Molecule Object, or zero on error.

IXA_MOL_GetNumBonds

```
int IXA_MOL_GetNumBonds(IXA_STATUS_HANDLE hStatus,
                        IXA_MOL_HANDLE hMolecule)
```

Description

Returns the total number of bonds in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

Output

The total number of bonds in the IXA Molecule Object, or zero on error.

IXA_MOL_GetAtomId

```
IXA_ATOMID IXA_MOL_GetAtomId(IXA_STATUS_HANDLE hStatus,
                             IXA_MOL_HANDLE hMolecule,
                             int vAtomIndex)
```

Description

Returns the IXA Atom Identifier for an atom in an IXA Molecule Object. This function provides a means for obtaining the IXA Atom Identifier for an atom, given its sequential index within the IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtomIndex: Index (from zero) of an atom in the IXA Molecule Object.

Output

IXA Atom Identifier for the specified atom in the specified IXA Molecule Object, or IXA_ATOMID_INVALID on error.

IXA_MOL_GetBondId

```
IXA_BONDID IXA_MOL_GetBondId(IXA_STATUS_HANDLE hStatus,
                             IXA_MOL_HANDLE hMolecule,
                             int vBondIndex)
```

Description

Returns the IXA Bond Identifier for a bond in an IXA Molecule Object. This function provides a means for obtaining the IXA Bond Identifier for a bond, given its sequential index within the IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vBondIndex: Index (from zero) of a bond in the IXA Molecule Object.

Output

IXA Bond Identifier for the specified bond in the specified Molecule, or IXA_BONDID_INVALID on error.

IXA_MOL_GetAtomIndex

```
int IXA_MOL_GetAtomIndex(IXA_STATUS_HANDLE hStatus,
                         IXA_MOL_HANDLE hMolecule,
                         IXA_ATOMID vAtom)
```

Description

Returns the index (from zero) for an atom (specified by IXA Atom Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for an atom in the IXA Molecule Object.

Output

The index (from zero) of the specified atom in the specified IXA Molecule Object, or zero on error.

IXA_MOL_GetBondIndex

```
int IXA_MOL_GetBondIndex(IXA_STATUS_HANDLE hStatus,
                         IXA_MOL_HANDLE hMolecule,
                         IXA_BONDID vBond)
```

Description

Returns the index (from zero) for a bond (specified by an IXA Bond Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vBond: IXA Bond Identifier for a bond in the IXA Molecule Object.

Output

The index (from zero) of the specified bond in the specified molecule, or zero on error.

IXA_MOL_GetAtomNumBonds

```
int IXA_MOL_GetAtomNumBonds(IXA_STATUS_HANDLE hStatus,
                            IXA_MOL_HANDLE hMolecule,
                            IXA_ATOMID vAtom)
```

Description

Returns the number of bonds attached to an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

Output

The number of bonds attached to the specified atom, or zero on error.

IXA_MOL_GetAtomBond

```
IXA_BONDID IXA_MOL_GetAtomBond(IXA_STATUS_HANDLE hStatus,
                               IXA_MOL_HANDLE hMolecule,
                               IXA_ATOMID vAtom,
                               int vBondIndex)
```

Description

Returns the IXA Bond Identifier for one of the bonds attached to an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

vBondIndex: The index (in the range zero to one less that the number of bonds attached to

vAtom – i.e. the value returned by IXA_MOL_GetAtomNumBonds) for the bond whose Identifier is to be returned.

Output

The IXA Bond Identifier for the specified bond, or IXA_BONDID_INVALID on error.

IXA_MOL_GetCommonBond

```
IXA_BONDID IXA_MOL_GetCommonBond(IXA_STATUS_HANDLE hStatus,
                                 IXA_MOL_HANDLE hMolecule,
                                 IXA_ATOMID vAtom1,
                                 IXA_ATOMID vAtom2)
```

Description

Returns the IXA Bond Identifier for the bond which joins two atoms in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom1: IXA Atom Identifier for the atom at one end of the bond.

vAtom2: IXA Atom Identifier for the atom at the other end of the bond.

Output

The IXA Bond Identifier for the bond which joins the two atoms, or IXA_BONDID_INVALID if no such bond exists, or on error.

IXA_MOL_GetBondAtom1

```
IXA_ATOMID IXA_MOL_GetBondAtom1(IXA_STATUS_HANDLE hStatus,
                                IXA_MOL_HANDLE hMolecule,
                                IXA_BONDID vBond)
```

Description

Returns the IXA Atom Identifier for the first atom involved in a specified bond in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vBond: IXA Bond Identifier for a bond in the IXA Molecule Object.

Output

IXA Atom Identifier for the first atom involved in the specified bond, or IXA_ATOMID_INVALID on error.

IXA_MOL_GetBondAtom2

```
IXA_ATOMID IXA_MOL_GetBondAtom2(IXA_STATUS_HANDLE hStatus,
                                IXA_MOL_HANDLE hMolecule,
                                IXA_BONDID vBond)
```

Description

Returns the IXA Atom Identifier for the second atom involved in a specified bond in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vBond: IXA Bond Identifier for a bond in the IXA Molecule Object.

Output

IXA Atom Identifier for the second atom involved in the specified bond, or IXA_ATOMID_INVALID on error.

IXA_MOL_GetBondOtherAtom (new in v. 1.06)

```
IXA_ATOMID IXA_MOL_GetBondOtherAtom(IXA_STATUS_HANDLE hStatus,
                                    IXA_MOL_HANDLE hMolecule,
                                    IXA_BONDID vBond,
                                    IXA_ATOMID vAtom)
```

Description

Returns the IXA Atom Identifier for another atom involved in a specified bond with specified atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vBond: IXA Bond Identifier for a bond in the IXA Molecule Object.

vAtom: IXA Atom Identifier for the one atom of a specified bond.

Output

IXA Atom Identifier for another atom involved in the specified bond, or IXA_ATOMID_INVALID on error.

Functions to Return Information About Atoms

IXA_MOL_GetAtomElement

```
const char* IXA_MOL_GetAtomElement(IXA_STATUS_HANDLE hStatus,
                                   IXA_MOL_HANDLE hMolecule,
                                   IXA_ATOMID vAtom)
```

Description

Returns the element type for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

Output

The IUPAC element symbol for the specified atom, or NULL on error. The returned string is owned by the IXA Molecule Object, and must be copied by the user if it is to be retained.

IXA_MOL_GetAtomAtomicNumber

```
int IXA_MOL_GetAtomAtomicNumber(IXA_STATUS_HANDLE hStatus,
                                IXA_MOL_HANDLE hMolecule,
                                IXA_ATOMID vAtom)
```

Description

Returns the atomic number for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

Output

The atomic number for the specified atom, or zero on error.

IXA_MOL_GetAtomMass

```
int IXA_MOL_GetAtomMass(IXA_STATUS_HANDLE hStatus,
                        IXA_MOL_HANDLE hMolecule,
                        IXA_ATOMID vAtom)
```

Description

Returns the mass number for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

Output

The mass number for the specified atom. The constant IXA_ATOM_NATURAL_MASS indicates the naturally-abundant mixture of masses, and zero is returned on error.

IXA_MOL_GetAtomCharge

```
int IXA_MOL_GetAtomCharge(IXA_STATUS_HANDLE hStatus,
                          IXA_MOL_HANDLE hMolecule,
                          IXA_ATOMID vAtom)
```

Description

Returns the formal charge on an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

Output

The formal charge on the specified atom, or zero on error.

IXA_MOL_GetAtomRadical

```
IXA_ATOM_RADICAL IXA_MOL_GetAtomRadical(IXA_STATUS_HANDLE hStatus,
                                        IXA_MOL_HANDLE hMolecule,
                                        IXA_ATOMID vAtom)
```

Description

Returns the radical state of an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

Output

The radical state constant value for the specified atom, or IXA_ATOM_RADICAL_NONE on error.

IXA_MOL_GetAtomHydrogens

```
int IXA_MOL_GetAtomHydrogens(IXA_STATUS_HANDLE hStatus,
                             IXA_MOL_HANDLE hMolecule,
                             IXA_ATOMID vAtom,
                             int vHydrogenMassNumber)
```

Description

Returns the number of hydrogen atoms of a specified mass which are attached to an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

vHydrogenMassNumber: The mass number for the hydrogen atoms of interest (in the range 1-3).

Output

The number of hydrogen atoms of the specified mass which are attached to the specified atom, or zero on error.

IXA_MOL_GetAtomX

```
double IXA_MOL_GetAtomX(IXA_STATUS_HANDLE hStatus,
                        IXA_MOL_HANDLE hMolecule,
                        IXA_ATOMID vAtom)
```

Description

Returns the *x*-coordinate for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

Output

x-coordinate for the specified atom, or zero on error.

IXA_MOL_GetAtomY

```
double IXA_MOL_GetAtomY(IXA_STATUS_HANDLE hStatus,
                        IXA_MOL_HANDLE hMolecule,
                        IXA_ATOMID vAtom)
```

Description

Returns the *y*-coordinate for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

Output

y-coordinate for the specified atom, or zero on error.

IXA_MOL_GetAtomZ

```
double IXA_MOL_GetAtomZ(IXA_STATUS_HANDLE hStatus,
                        IXA_MOL_HANDLE hMolecule,
                        IXA_ATOMID vAtom)
```

Description

Returns the *z*-coordinate for an atom in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vAtom: IXA Atom Identifier for the atom to be examined.

Output

z-coordinate for the specified atom, or zero on error.

Functions to Return Information About Bonds

IXA_MOL_GetBondType

```
IXA_BOND_TYPE IXA_MOL_GetBondType(IXA_STATUS_HANDLE hStatus,
                                  IXA_MOL_HANDLE hMolecule,
                                  IXA_BONDID vBond)
```

Description

Returns the bond type for a bond in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vBond: IXA Bond Identifier for the bond to be examined.

Output

The bond type for the specified bond, or IXA_BOND_TYPE_SINGLE on error.

IXA_MOL_GetBondWedge

```
IXA_BOND_WEDGE IXA_MOL_GetBondWedge(IXA_STATUS_HANDLE hStatus,
                                    IXA_MOL_HANDLE hMolecule,
                                    IXA_BONDID vBond,
                                    IXA_ATOMID vRefAtom)
```

Description

Returns the wedge direction for a bond in an IXA Molecule Object with respect to a specified atom. Note that the wedge direction is defined only for the reference atom; i.e. if this function is called on the atoms at both ends of a bond, the fact that it returns IXA_BOND_WEDGE_UP for one atom does not imply that it will return IXA_BOND_WEDGE_DOWN for the other.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vBond: IXA Bond Identifier for the bond to be examined.

vRefAtom: IXA Atom Identifier for the reference atom, at one end of the specified bond.

Output

The wedge direction for the specified bond from the specified atom.

IXA_MOL_GetDblBondConfig

```
IXA_DBLBOND_CONFIG IXA_MOL_GetDblBondConfig(IXA_STATUS_HANDLE hStatus,
                                            IXA_MOL_HANDLE hMolecule,
                                            IXA_BONDID vBond)
```

Description

Returns the stereo configuration for a double bond in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vBond: IXA Bond Identifier for the bond to be examined.

Output

The double bond configuration for the specified bond.

Functions to Return Information About Stereodescriptors

IXA_MOL_GetNumStereos

```
int IXA_MOL_GetNumStereos(IXA_STATUS_HANDLE hStatus,
                          IXA_MOL_HANDLE hMolecule)
```

Description

Returns the total number of stereodescriptors in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

Output

The total number of stereodescriptors in the IXA Molecule Object.

IXA_MOL_GetStereoId

```
IXA_STEREOID IXA_MOL_GetStereoId(IXA_STATUS_HANDLE hStatus,
                                 IXA_MOL_HANDLE hMolecule,
                                 int vStereoIndex)
```

Description

Returns the IXA Stereodescriptor Identifier for a stereodescriptor in an IXA Molecule Object. This function provides a means for obtaining the IXA Stereodescriptor Identifier for a stereodescriptor, given its sequential index within the IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vStereoIndex: Index (from zero) of a stereodescriptor in the IXA Molecule Object.

Output

IXA Stereodescriptor Identifier for the specified stereodescriptor in the specified IXA Molecule Object, or IXA_STEREOID_INVALID on error.

IXA_MOL_GetStereoIndex

```
int IXA_MOL_GetStereoIndex(IXA_STATUS_HANDLE hStatus,
                           IXA_MOL_HANDLE hMolecule,
                           IXA_STEREOID vStereo)
```

Description

Returns the index (from zero) for a stereodescriptor (specified by an IXA Stereodescriptor Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vStereo: IXA Stereodescriptor Identifier for a stereodescriptor in the IXA Molecule Object.

Output

The index (from zero) of the specified stereodescriptor in the specified molecule, or zero on error.

IXA_MOL_GetStereoTopology

```
IXA_STEREO_TOPOLOGY IXA_MOL_GetStereoTopology(IXA_STATUS_HANDLE hStatus,
                                              IXA_MOL_HANDLE hMolecule,
                                              IXA_STEREOID vStereo)
```

Description

Returns the topology of a stereodescriptor (specified by an IXA Stereodescriptor Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vStereo: IXA Stereodescriptor Identifier for a stereodescriptor in the IXA Molecule Object.

Output

The topology of the specified stereodescriptor in the specified molecule, or IXA_MOL_STEREOTOPOLOGY_INVALID on error.

IXA_MOL_GetStereoCentralAtom

```
IXA_ATOMID IXA_MOL_GetStereoCentralAtom(IXA_STATUS_HANDLE hStatus,
                                        IXA_MOL_HANDLE hMolecule,
                                        IXA_STEREOID vStereo)
```

Description

Returns the IXA Atom Identifier for the central atom of a stereodescriptor (specified by an IXA Stereodescriptor Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vStereo: IXA Stereodescriptor Identifier for the stereodescriptor to be examined.

Output

IXA Atom Identifier for the central atom of the specified stereodescriptor in the specified IXA Molecule Object, or IXA_ATOMID_INVALID on error.

IXA_MOL_GetStereoCentralBond

```
IXA_BONDID IXA_MOL_GetStereoCentralBond(IXA_STATUS_HANDLE hStatus,
                                        IXA_MOL_HANDLE hMolecule,
                                        IXA_STEREOID vStereo)
```

Description

Returns the IXA Bond Identifier for the central bond of a stereodescriptor (specified by an IXA Stereodescriptor Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vStereo: IXA Stereodescriptor Identifier for the stereodescriptor to be examined.

Output

IXA Bond Identifier for the central bond of the specified stereodescriptor in the specified IXA Molecule Object, or IXA_BONDID_INVALID on error.

IXA_MOL_GetStereoNumVertices

```
int IXA_MOL_GetStereoNumVertices (IXA_STATUS_HANDLE hStatus,
                                  IXA_MOL_HANDLE hMolecule,
                                  IXA_STEREOID vStereo)
```

Description

Returns the number of vertices involved in a stereodescriptor (specified by an IXA Stereodescriptor Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vStereo: IXA Stereodescriptor Identifier for a stereodescriptor in the IXA Molecule Object.

Output

The number of vertices involved in the specified stereodescriptor in the specified IXA Molecule Object, or zero on error.

IXA_MOL_GetStereoVertex

```
IXA_ATOMID IXA_MOL_GetStereoVertex(IXA_STATUS_HANDLE hStatus,
                                   IXA_MOL_HANDLE hMolecule,
                                   IXA_STEREOID vStereo,
                                   int vVertexIndex)
```

Description

Returns the IXA Atom Identifier for one of the vertices involved in a stereodescriptor (specified by an IXA Stereodescriptor Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vStereo: IXA Stereodescriptor Identifier for a stereodescriptor in the IXA Molecule

Object.

vVertexIndex: Index number (from zero) for the vertex whose IXA Atom Identifier is required.

Output

IXA Atom Identifier for the specified vertex in the specified stereodescriptor in the specified IXA Molecule Object, or IXA_ATOMID_INVALID on error.

IXA_MOL_GetStereoParity

```
IXA_STEREO_PARITY IXA_MOL_GetStereoParity(IXA_STATUS_HANDLE hStatus,
                                          IXA_MOL_HANDLE hMolecule,
                                          IXA_STEREOID vStereo)
```

Description

Returns the parity value for a stereodescriptor (specified by IXA Stereodescriptor Identifier) in an IXA Molecule Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hMolecule: Handle for the IXA Molecule Object to be examined.

vStereo: IXA Stereodescriptor Identifier for the stereodescriptor to be examined.

Output

The parity value for the specified stereodescriptor in the specified IXA Molecule Object.

InChI Builder Objects

IXA InChI Builder Objects are used to generate InChIs and Auxiliary Data for the molecules represented in IXA Molecule Objects. The basic procedure is to associate an IXA Molecule Object with an IXA InChI Builder Object, set any options required, and then extract the InChI from it, along with Auxiliary Data and Log Data, if required. By default (if no options are specified) a standard InChI is generated. The actual process of InChI generation occurs when the first function call is made to extract the InChI, Auxiliary Data or Log Data, for a particular associated IXA Molecule Object and set of InChI-generation options.

Types and Constants

IXA InChI Builder Objects have Handles of type IXA_INCHIBUILDER_HANDLE. Most options controlling InChI generation are on/off switches. The switches are referenced as constants of type IXA_INCHIBUILDER_OPTION, as follows:

- IXA_INCHIBUILDER_OPTION_NewPsOff: If set to IXA_FALSE, only the narrow end of a stereochemistry wedge bond points to a stereocentre (Standard InChI); if set to IXA_TRUE, both ends of a stereochemistry wedge bond point to stereocentres.

- IXA_INCHIBUILDER_OPTION_DoNotAddH: If set to IXA_FALSE, hydrogens are added to nonhydrogen atoms according to normal valences (Standard InChI); if set to IXA_TRUE, all hydrogens in the IXA Molecule must be specified explicitly, either by adding them as separate atoms, or by specifying them using function IXA_MOL_SetAtomHydrogens.

- IXA_INCHIBUILDER_OPTION_SUU: ("Stereo Unknown Undefined") If set to IXA_FALSE, unknown or undefined stereochemistry is not indicated unless at least one defined stereocentre is present (Standard InChI); if set to IXA_TRUE, unknown or undefined stereochemistry is always indicated.

- IXA_INCHIBUILDER_OPTION_SLUUD: ("Stereo Labels for Unknown and Undefined are Different") If set to IXA_FALSE, the stereo labels for both unknown and undefined stereocentres are shown as "?" (Standard InChI); if set to IXA_TRUE, the stereo labels for unknown stereo-chemistry are shown as "u", while those for undefined are shown as "?".

• IXA_INCHIBUILDER_OPTION_FixedH: If set to IXA_FALSE, no Fixed H layer is included (Standard InChI); if set to IXA_TRUE, a Fixed H layer is included.

• IXA_INCHIBUILDER_OPTION_RecMet: If set to IXA_FALSE, reconnected metals results are not included (Standard InChI); If set to IXA_TRUE, reconnected metals results are included.

• IXA_INCHIBUILDER_OPTION_KET: ("Keto-Enol Tautomerism") If set to IXA_FALSE, keto-enol tautomerism is ignored (Standard InChI); if set to IXA_TRUE, keto-enol tautomerism is accounted for (experimental extension to InChI 1).

• IXA_INCHIBUILDER_OPTION_15T ("1,5-Tautomerism") If set to IXA_FALSE, 1,5-tautomerism is ignored (Standard InChI); if set to IXA_TRUE, 1,5-tautomerism is accounted for (experimental extension to InChI 1).

• IXA_INCHIBUILDER_OPTION_SaveOpt: If set to IXA_FALSE, any options used for non-standard InChI generation are not saved in the InChI string; if set to IXA_TRUE, any options used for nonstandard InChI generation are saved in the InChI string.

• IXA_INCHIBUILDER_OPTION_AuxNone: If set to IXA_FALSE, auxiliary information is generated alongside the InChI (default); if set to IXA_TRUE, no auxiliary information is generated.

• IXA_INCHIBUILDER_OPTION_WarnOnEmptyStructure: If set to IXA_FALSE (default), no warning is generated if an empty structure (IXA Molecule Object with zero atoms) is used to generate an InChI; if set to IXA_TRUE,a warning message is added to the IXA Status Object, and an empty InChI is generated.

• IXA_INCHIBUILDER_OPTION_Polymers: If set to IXA_FALSE (default), no polymer treatment is allowed; if set to IXA_TRUE, polymers are handled.

• IXA_INCHIBUILDER_OPTION_Polymers105: If set to IXA_TRUE, polymers are handled in legacy (v. 1.05) mode

• IXA_INCHIBUILDER_OPTION_NPZZ: If set to IXA_FALSE (default), non-polymeric ZZ atoms are disabled; if set to IXA_TRUE, non-polymeric ZZ atoms are allowed.

• IXA_INCHIBUILDER_OPTION_NoFrameShift: If set to IXA_FALSE (default), attempt is made to canonicalize CRU by frame; if set to IXA_TRUE, frame shift is disabled.

• IXA_INCHIBUILDER_OPTION_FoldCRU: If set to IXA_FALSE (default), no polymer CRU folding is attempted; if set to IXA_TRUE, attempt is made to fold CRU in order to eliminate inner repeats (e.g., convert -(CH2CH2)n- to -(CH2)n-).

• IXA_INCHIBUILDER_OPTION_LooseTSACheck: If set to IXA_FALSE (default), usual strict criteria of ambiguous drawing for in-ring tetrahedral stereo are used;; if set to IXA_TRUE, relaxed critera are used (useful for large rings where in-ring bond angles are close to 180o).

• IXA_INCHIBUILDER_OPTION_OutErrInChI: If set to IXA_FALSE (default), no InChI output occurs on error;if set to IXA_TRUE, empty InChI ("InChI=1//" or "InChI=1S//" string is produced.

• IXA_INCHIBUILDER_OPTION_NoWarnings: If set to IXA_FALSE (default), warnings are prouced as usual;if set to IXA_TRUE, output of warnings is suppressed.

Options for the interpretation of stereochemistry during InChI generation are constants of type IXA_INCHIBUILDER_STEREOOPTION, as follows:

• IXA_INCHIBUILDER_STEREOOPTION_SAbs (use absolute stereochemistry - this is the default option and allows a Standard InChI to be generated)

• IXA_INCHIBUILDER_STEREOOPTION_SNon ignore all stereochemistry)

- IXA_INCHIBUILDER_STEREOOPTION_SRel (use relative stereochemistry)

- IXA_INCHIBUILDER_STEREOOPTION_SRac (use racemic stereochemistry)

- IXA_INCHIBUILDER_STEREOOPTION_SUCF (use the chiral flag set for the IXA Molecule Object by function IXA_MOL_SetChiral to determine how to interpret stereochemistry: use absolute stereochemistry if the chiral flag is IXA_TRUE; use relative stereochemistry if it is IXA_FALSE)

Functions to Generate InChIs

IXA_INCHIBUILDER_Create

```
IXA_INCHIBUILDER_HANDLE IXA_INCHIBUILDER_Create(IXA_STATUS_HANDLE hStatus)
```

Description

Creates a new empty IXA InChI Builder Object and returns its handle.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

Output

Handle for the newly-created IXA InChI Builder Object.

IXA_INCHIBUILDER_SetMolecule

```
void IXA_INCHIBUILDER_SetMolecule(IXA_STATUS_HANDLE hStatus,
                                  IXA_INCHIBUILDER_HANDLE hBuilder,
                                  IXA_MOL_HANDLE hMolecule)
```

Description

Associates an IXA Molecule Object with an IXA InChI Builder Object, replacing any IXA Molecule Object previously associated with it.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object to be modified.

hMolecule: Handle for the IXA Molecule Object to be associated with the IXA InChI Builder Object.

IXA_INCHIBUILDER_GetInChI

```
const char* IXA_INCHIBUILDER_GetInChI(IXA_STATUS_HANDLE hStatus,
                                      IXA_INCHIBUILDER_HANDLE hBuilder)
```

Description

Returns a string containing the InChI for the molecule described in the IXA Molecule Object currently associated with an IXA InChI Builder Object, based on any options currently set for the IXA InChI Builder Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object to be examined.

Output

Null-terminated string containing the InChI for the IXA Molecule Object currently associated with the IXA InChI Builder Object; NULL is returned on error. The returned string is owned by the IXA InChI Builder Object, and is liable to change if the IXA InChI Builder Object, or the IXA Molecule Object associated with it, is modified in any way. The string must therefore be copied by the user if it is to be retained.

Note

Since v. 1.06, this function provides full-scale (though experimental) support of polymers. This requires specifying option Polymers" (or "Polymers105" to request older v. 1.05 compatibility mode) via corresponding call to IXA_INCHIBUILDER_SetOption.

IXA_INCHIBUILDER_GetAuxInfo

```
const char* IXA_INCHIBUILDER_GetAuxInfo(IXA_STATUS_HANDLE hStatus,
                                        IXA_INCHIBUILDER_HANDLE hBuilder)
```

Description

Returns a string containing the Auxiliary Information for the molecule described in the IXA Molecule Object currently associated with an IXA InChI Builder Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object to be examined.

Output

Null-terminated string containing the Auxiliary Information for molecule described in the IXA Molecule Object currently associated with the IXA InChI Builder Object. NULL is returned on error. The returned string is owned by the IXA InChI Builder Object, and is liable to change if the IXA InChI Builder Object, or the IXA Molecule Object associated with it, is modified in any way. The string must therefore be copied by the user if it is to be retained.

IXA_INCHIBUILDER_GetLog

const char* IXA_INCHIBUILDER_GetLog

(IXA_STATUS_HANDLE hStatus,

IXA_INCHIBUILDER_HANDLE hBuilder)

Description

Returns a string containing Log Data for the generation of the InChI for the molecule described in the IXA Molecule Object currently associated with an IXA InChI Builder Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object to be examined.

Output

Null-terminated string containing Log Data for the generation of the InChI for the molecule described in the IXA Molecule Object currently associated with the IXA InChI Builder Object.

NULL is returned on error. The returned string is owned by the IXA InChI Builder Object, and is liable to change if the IXA InChI Builder Object, or the IXA Molecule Object associated with it, is modified in any way. The string must therefore be copied by the user if it is to be retained.

IXA_INCHIBUILDER_Destroy

```
void IXA_INCHIBUILDER_Destroy(IXA_STATUS_HANDLE hStatus,
                              IXA_INCHIBUILDER_HANDLE hBuilder)
```

Description

Destroys an IXA InChI Builder Object, releasing all memory that it uses.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object to be destroyed.

Functions to Set InChI-Generation Options

The functions described in this section allow generation of non-standard InChIs by specifying various nonstandard options; in addition, a processing timeout can be imposed on the actual generation of the InChI.

IXA_INCHIBUILDER_SetOption

```
void IXA_INCHIBUILDER_SetOption(IXA_STATUS_HANDLE hStatus,
                                IXA_INCHIBUILDER_HANDLE hBuilder,
                                IXA_INCHIBUILDER_OPTION vOption,
                                IXA_BOOL vValue)
```

Description

Sets an “on/off” option for InChI generation using an IXA InChI Builder Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object for which the option is to be set. vOption: InChI generation option to be set.

Valid options are:

IXA_INCHIBUILDER_OPTION_NewPsOff

IXA_INCHIBUILDER_OPTION_DoNotAddH

IXA_INCHIBUILDER_OPTION_SUU

IXA_INCHIBUILDER_OPTION_SLUUD

IXA_INCHIBUILDER_OPTION_FixedH

IXA_INCHIBUILDER_OPTION_RecMet

IXA_INCHIBUILDER_OPTION_KET

IXA_INCHIBUILDER_OPTION_15T

IXA_INCHIBUILDER_OPTION_SaveOpt

IXA_INCHIBUILDER_OPTION_AuxNone

IXA_INCHIBUILDER_OPTION_WarnOnEmptyStructure

IXA_INCHIBUILDER_OPTION_Polymers

IXA_INCHIBUILDER_OPTION_Polymers105

IXA_INCHIBUILDER_OPTION_NoFrameShift

IXA_INCHIBUILDER_OPTION_NPZZ

IXA_INCHIBUILDER_OPTION_FoldCRU

IXA_INCHIBUILDER_OPTION_LooseTSACheck

IXA_INCHIBUILDER_OPTION_NoWarnings

IXA_INCHIBUILDER_OPTION_OutErrInChI
IXA_INCHIBUILDER_OPTION_LargeMolecules

vValue: Value to be used for the specified option. IXA_TRUE means that the specified option should be applied; IXA_FALSE means that the option should not be applied, and is the default situation if this function is not called at all for the IXA InChI Builder Object. If all options are set to IXA_FALSE, a Standard InChI is generated.

IXA_INCHIBUILDER_SetOption_Stereo

```
void IXA_INCHIBUILDER_SetOption_Stereo(IXA_STATUS_HANDLE hStatus,
                                       IXA_INCHIBUILDER_HANDLE hBuilder,
                                       INCHIBUILDER_STEREOOPTION vValue)
```

Description

Sets an option for interpretation of stereochemistry for InChI generation. If this function is not called to set an option, the default option is to use absolute stereochemistry (INCHIBUILDER_STEREOOPTION_SAbs), which generates a Standard InChI.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object for which the option is to be set.

vValue: Option value to be applied for interpretation of stereochemistry in InChI generation.

IXA_INCHIBUILDER_SetOption_Timeout

```
void IXA_INCHIBUILDER_SetOption_Timeout(IXA_STATUS_HANDLE hStatus,
                                        IXA_INCHIBUILDER_HANDLE hBuilder,
                                        int vValue)
```

Description

Sets a timeout for InChI generation in seconds. Functions which involve the generation of InChIs will fail if the specified timeout is exceeded.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object whose behaviour is to be modified. vValue: Maximum time permitted in seconds. A value of zero indicates that no timeout is applied, and is the default if this function is never called.

IXA_INCHIBUILDER_SetOption_Timeout_Milliseconds (new in v. 1.06)

```
void IXA_INCHIBUILDER_SetOption_Timeout_MilliSeconds(IXA_STATUS_HANDLE hStatus,
                                                     IXA_INCHIBUILDER_HANDLE
hBuilder,
                                                     long vValue)
```

Description

Sets a timeout for InChI generation in milliseconds (useful for performing mass generation of InChI for small molecules, as well as testing). Functions which involve the generation of InChIs will fail if the specified timeout is exceeded.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object whose behaviour is to be modified. vValue: Maximum time permitted in milliseconds. A value of zero indicates that no timeout is applied, and is the default if this function is never called.

IXA_INCHIBUILDER_CheckOption (new in v. 1.06)

```
IXA_BOOL IXA_INCHIBUILDER_CheckOption(IXA_STATUS_HANDLE hStatus,
                                      IXA_INCHIBUILDER_HANDLE hBuilder,
                                      IXA_INCHIBUILDER_OPTION vOption)
```

Description

Checks if an option for InChI generation is set to “on/off”.

Returns IXA_TRUE for “on” or IXA_FALSE for “off”.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object for which the option is to be set.

vOption: InChI generation option to check (see IXA_INCHIBUILDER_SetOption).

IXA_INCHIBUILDER_CheckOption_Stereo (new in v. 1.06)

```
IXA_BOOL IXA_INCHIBUILDER_CheckOption_Stereo(IXA_STATUS_HANDLE hStatus,
                                             IXA_INCHIBUILDER_HANDLE hBuilder,
                                             INCHIBUILDER_STEREOOPTION vValue)
```

Description

Checks if an option for interpretation of stereochemistry is equal to the specific value.

Returns either IXA_TRUE or IXA_FALSE.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object for which the option is to be set.

vValue: Option value to compare with that specified by IXA InChI Builder Object.

IXA_INCHIBUILDER_IXA_INCHIBUILDER_GetOption_Timeout_MilliSeconds (new in v. 1.06)

```
long IXA_INCHIBUILDER_GetOption_Timeout_MilliSeconds(IXA_STATUS_HANDLE hStatus,
                                                     IXA_INCHIBUILDER_HANDLE
hBuilder)
```

Description

Returns the value of timeout per molecule in milliseconds (which has been set by call of either IXA_INCHIBUILDER_SetOption_Timeout_MilliSeconds or IXA_INCHIBUILDER_SetOption_Timeout).

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hBuilder: Handle for the IXA InChI Builder Object for which the option is to be set.

InChIKey Builder Objects

IXA InChIKey Builder Objects are used for the generation of InChIKeys. The basic procedure is to associate an InChI with the IXA InChIKey Builder Object, and then extract the corresponding InChIKey from it. IXA InChIKey Builder Objects have Handles of type IXA_INCHIKEYBUILDER_HANDLE.

IXA_INCHIKEYBUILDER_Create

```
IXA_INCHIKEYBUILDER_HANDLE IXA_INCHIKEYBUILDER_Create(IXA_STATUS_HANDLE hStatus)
```

Description

Creates a new IXA InChIKey Builder Object and returns its Handle.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

Output

Handle for the newly-created IXA InChIKey Builder Object.

IXA_INCHIKEYBUILDER_SetInChI

```
void IXA_INCHIKEYBUILDER_SetInChI(IXA_STATUS_HANDLE hStatus,
                                  IXA_INCHIKEYBUILDER_HANDLE hInChIKeyBuilder,
                                  const char* pInChI)
```

Description

Associates an InChI with an IXA InChIKey Builder Object, replacing any InChI previously associated with it.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hInChIKeyBuilder: Handle for the IXA InChIKey Builder Object to be modified.

pInChI: Null-terminated character string containing the InChI to be associated with the IXA InChIKey Builder Object.

IXA_INCHIKEYBUILDER_GetInChIKey

```
const char* IXA_INCHIKEYBUILDER_GetInChIKey(IXA_STATUS_HANDLE hStatus,
                                            IXA_INCHIKEYBUILDER_HANDLE
hInChIKeyBuilder)
```

Description

Returns a string containing the InChIKey corresponding to the InChI currently associated with an IXA InChIKey Builder Object.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hInChIKeyBuilder: Handle for the IXA InChIKey Builder Object to be used for InChIKey generation.

Output

Null-terminated string containing the InChIKey for the InChI currently associated with the IXA InChIKey Builder Object. The returned string is owned by the IXA InChIKey Builder Object, and is liable to change if the IXA InChIKey Builder Object is modified in any way. The string must therefore be copied by the user if it is to be retained.

IXA_INCHIKEYBUILDER_Destroy

```
void IXA_INCHIKEYBUILDER_Destroy(IXA_STATUS_HANDLE hStatus,
                                 IXA_INCHIKEYBUILDER_HANDLE hInChIKeyBuilder)
```

Description

Destroys an IXA InChIKey Builder Object, releasing all memory that it uses.

Input

hStatus: Handle for an IXA Status Object to receive status messages.

hInChIKeyBuilder: Handle for the IXA InChIKey Builder Object to be destroyed.