**IUPAC International Chemical Identifier (InChI)**

**InChI version 1, Software version 1.06 pre-release PR2**

**API Reference**

Last revision date: August 17, 2020

This document is a part of the release of the IUPAC International Chemical Identifier with InChIKey, version 1, software version 1.06 pre-release PR2.

**CONTENTS**

[Overview 8](#_Toc31395418)

[Classic InChI API 9](#_Toc31395419)

[Generation of InChI from structure 9](#_Toc31395420)

[GetINCHI 9](#_Toc31395421)

[GetINCHIEx 13](#_Toc31395422)

[FreeINCHI 15](#_Toc31395423)

[Free\_inchi\_Input 15](#_Toc31395424)

[Get\_inchi\_Input\_FromAuxInfo 15](#_Toc31395425)

[GetStdINCHI 17](#_Toc31395426)

[FreeStdINCHI 18](#_Toc31395427)

[Free\_std\_inchi\_Input 18](#_Toc31395428)

[Get\_std\_inchi\_Input\_FromAuxInfo 18](#_Toc31395429)

[Generation of InChI from structure, step-by-step way 18](#_Toc31395430)

[INCHIGEN\_Create 19](#_Toc31395431)

[INCHIGEN\_Setup 20](#_Toc31395432)

[INCHIGEN\_DoNormalization 21](#_Toc31395433)

[INCHIGEN\_DoCanonicalization 21](#_Toc31395434)

[INCHIGEN\_DoSerialization 22](#_Toc31395435)

[INCHIGEN\_Reset 23](#_Toc31395436)

[INCHIGEN\_Destroy 24](#_Toc31395437)

[STDINCHIGEN\_Create 24](#_Toc31395438)

[STDINCHIGEN\_Setup 24](#_Toc31395439)

[STDINCHIGEN\_DoNormalization 25](#_Toc31395440)

[STDINCHIGEN\_DoCanonicalization 26](#_Toc31395441)

[STDINCHIGEN\_DoSerialization 26](#_Toc31395442)

[STDINCHIGEN\_Reset 27](#_Toc31395443)

[STDINCHIGEN\_Destroy 27](#_Toc31395444)

[Generation of InChI directly from Molfile 28](#_Toc31395445)

[MakeINCHIFromMolfileText 28](#_Toc31395446)

[Restoring structure from InChI or AuxInfo 29](#_Toc31395447)

[GetStructFromINCHI 29](#_Toc31395448)

[GetStructFromINCHIEx 29](#_Toc31395449)

[FreeStructFromINCHI 30](#_Toc31395450)

[GetStructFromStdINCHI 30](#_Toc31395451)

[FreeStructFromStdINCHI 31](#_Toc31395452)

[InChIKey 31](#_Toc31395453)

[GetINCHIKeyFromINCHI 31](#_Toc31395454)

[CheckINCHIKey 33](#_Toc31395455)

[GetStdINCHIKeyFromStdINCHI 34](#_Toc31395456)

[Test and utlity procedures 34](#_Toc31395457)

[GetINCHIfromINCHI 34](#_Toc31395458)

[CheckINCHI 35](#_Toc31395459)

[GetStringLength 36](#_Toc31395460)

[InChI Extensible API – IXA 37](#_Toc31395461)

[Status Objects 38](#_Toc31395462)

[Types and Constants 38](#_Toc31395463)

[Functions 39](#_Toc31395464)

[IXA\_STATUS\_Create 39](#_Toc31395465)

[IXA\_STATUS\_Clear 39](#_Toc31395466)

[IXA\_STATUS\_Destroy 39](#_Toc31395467)

[IXA\_STATUS\_HasError 40](#_Toc31395468)

[XA\_STATUS\_HasWarning 40](#_Toc31395469)

[IXA\_STATUS\_GetCount 41](#_Toc31395470)

[IXA\_STATUS\_GetSeverity 41](#_Toc31395471)

[IXA\_STATUS\_GetMessage 42](#_Toc31395472)

[Molecule Objects 42](#_Toc31395473)

[Stereochemistry 43](#_Toc31395474)

[Types and Constants 43](#_Toc31395475)

[Functions to Create, Clear and Destroy Molecule Objects 46](#_Toc31395476)

[IXA\_MOL\_Create 46](#_Toc31395477)

[IXA\_MOL\_Clear 46](#_Toc31395478)

[IXA\_MOL\_Destroy 47](#_Toc31395479)

[Functions Operating on Complete Molecules 47](#_Toc31395480)

[IXA\_MOL\_ReadMolfile 47](#_Toc31395481)

[IXA\_MOL\_ReadInChI 48](#_Toc31395482)

[IXA\_MOL\_SetChiral 48](#_Toc31395483)

[IXA\_MOL\_GetChiral 49](#_Toc31395484)

[Functions to Add and Define Atoms 50](#_Toc31395485)

[IXA\_MOL\_CreateAtom 50](#_Toc31395486)

[IXA\_MOL\_SetAtomElement 51](#_Toc31395487)

[IXA\_MOL\_SetAtomAtomicNumber 51](#_Toc31395488)

[IXA\_MOL\_SetAtomMass 52](#_Toc31395489)

[IXA\_MOL\_SetAtomCharge 53](#_Toc31395490)

[IXA\_MOL\_SetAtomRadical 53](#_Toc31395491)

[IXA\_MOL\_SetAtomHydrogens 54](#_Toc31395492)

[IXA\_MOL\_SetAtomX 55](#_Toc31395493)

[IXA\_MOL\_SetAtomY 55](#_Toc31395494)

[IXA\_MOL\_SetAtomZ 56](#_Toc31395495)

[Functions to Add and Define Bonds 56](#_Toc31395496)

[IXA\_MOL\_CreateBond 56](#_Toc31395497)

[IXA\_MOL\_SetBondType 57](#_Toc31395498)

[IXA\_MOL\_SetBondWedge 58](#_Toc31395499)

[IXA\_MOL\_SetDblBondConfig 59](#_Toc31395500)

[Functions to Add and Define Stereodescriptors 59](#_Toc31395501)

[IXA\_MOL\_CreateStereoTetrahedron 60](#_Toc31395502)

[IXA\_MOL\_CreateStereoRectangle 61](#_Toc31395503)

[IXA\_MOL\_CreateStereoAntiRectangle 62](#_Toc31395504)

[IXA\_MOL\_SetStereoParity 64](#_Toc31395505)

[Functions to Add and Define Polymer Units 64](#_Toc31395506)

[IXA\_MOL\_CreatePolymerUnit (new in v. 1.06) 64](#_Toc31395507)

[IXA\_MOL\_GetPolymerUnitId (new in v. 1.06) 65](#_Toc31395508)

[IXA\_MOL\_GetPolymerUnitIndex (new in v. 1.06) 65](#_Toc31395509)

[IXA\_MOL\_SetPolymerUnit (new in v. 1.06) 65](#_Toc31395510)

[Functions to Navigate Within a Molecule 67](#_Toc31395511)

[IXA\_MOL\_GetNumAtoms 67](#_Toc31395512)

[IXA\_MOL\_GetNumBonds 68](#_Toc31395513)

[IXA\_MOL\_GetAtomId 68](#_Toc31395514)

[IXA\_MOL\_GetBondId 69](#_Toc31395515)

[IXA\_MOL\_GetAtomIndex 69](#_Toc31395516)

[IXA\_MOL\_GetBondIndex 70](#_Toc31395517)

[IXA\_MOL\_GetAtomNumBonds 71](#_Toc31395518)

[IXA\_MOL\_GetAtomBond 71](#_Toc31395519)

[IXA\_MOL\_GetCommonBond 72](#_Toc31395520)

[IXA\_MOL\_GetBondAtom1 73](#_Toc31395521)

[IXA\_MOL\_GetBondAtom2 73](#_Toc31395522)

[IXA\_MOL\_GetBondOtherAtom (new in v. 1.06) 74](#_Toc31395523)

[Functions to Return Information About Atoms 75](#_Toc31395524)

[IXA\_MOL\_GetAtomElement 75](#_Toc31395525)

[IXA\_MOL\_GetAtomAtomicNumber 75](#_Toc31395526)

[IXA\_MOL\_GetAtomMass 76](#_Toc31395527)

[IXA\_MOL\_GetAtomCharge 77](#_Toc31395528)

[IXA\_MOL\_GetAtomRadical 77](#_Toc31395529)

[IXA\_MOL\_GetAtomHydrogens 78](#_Toc31395530)

[IXA\_MOL\_GetAtomX 79](#_Toc31395531)

[IXA\_MOL\_GetAtomY 79](#_Toc31395532)

[IXA\_MOL\_GetAtomZ 80](#_Toc31395533)

[Functions to Return Information About Bonds 80](#_Toc31395534)

[IXA\_MOL\_GetBondType 80](#_Toc31395535)

[IXA\_MOL\_GetBondWedge 81](#_Toc31395536)

[IXA\_MOL\_GetDblBondConfig 82](#_Toc31395537)

[Functions to Return Information About Stereodescriptors 83](#_Toc31395538)

[IXA\_MOL\_GetNumStereos 83](#_Toc31395539)

[IXA\_MOL\_GetStereoId 83](#_Toc31395540)

[IXA\_MOL\_GetStereoIndex 84](#_Toc31395541)

[IXA\_MOL\_GetStereoTopology 84](#_Toc31395542)

[IXA\_MOL\_GetStereoCentralAtom 85](#_Toc31395543)

[IXA\_MOL\_GetStereoCentralBond 86](#_Toc31395544)

[IXA\_MOL\_GetStereoNumVertices 86](#_Toc31395545)

[IXA\_MOL\_GetStereoVertex 87](#_Toc31395546)

[IXA\_MOL\_GetStereoParity 88](#_Toc31395547)

[InChI Builder Objects 88](#_Toc31395548)

[Types and Constants 89](#_Toc31395549)

[Functions to Generate InChIs 91](#_Toc31395550)

[IXA\_INCHIBUILDER\_Create 91](#_Toc31395551)

[IXA\_INCHIBUILDER\_SetMolecule 91](#_Toc31395552)

[IXA\_INCHIBUILDER\_GetInChI 92](#_Toc31395553)

[IXA\_INCHIBUILDER\_GetAuxInfo 93](#_Toc31395554)

[IXA\_INCHIBUILDER\_GetLog 94](#_Toc31395555)

[IXA\_INCHIBUILDER\_Destroy 94](#_Toc31395556)

[Functions to Set InChI-Generation Options 95](#_Toc31395557)

[IXA\_INCHIBUILDER\_SetOption 95](#_Toc31395558)

[IXA\_INCHIBUILDER\_SetOption\_Stereo 96](#_Toc31395559)

[IXA\_INCHIBUILDER\_SetOption\_Timeout 97](#_Toc31395560)

[IXA\_INCHIBUILDER\_SetOption\_Timeout\_Milliseconds (new in v. 1.06) 98](#_Toc31395561)

[IXA\_INCHIBUILDER\_CheckOption (new in v. 1.06) 98](#_Toc31395562)

[IXA\_INCHIBUILDER\_CheckOption\_Stereo (new in v. 1.06) 99](#_Toc31395563)

[IXA\_INCHIBUILDER\_IXA\_INCHIBUILDER\_GetOption\_Timeout\_MilliSeconds (new in v. 1.06) 100](#_Toc31395564)

[InChIKey Builder Objects 101](#_Toc31395565)

[IXA\_INCHIKEYBUILDER\_Create 101](#_Toc31395566)

[IXA\_INCHIKEYBUILDER\_SetInChI 101](#_Toc31395567)

[IXA\_INCHIKEYBUILDER\_GetInChIKey 102](#_Toc31395568)

[IXA\_INCHIKEYBUILDER\_Destroy 103](#_Toc31395569)

## Overview

The current version of InChI Identifier is 1; the current status of the InChI software is 1.05 (Winter 2017) release. Previously released versions 1.01 (2006), 1.02-beta (2007), 1.02-standard (2009), 1.03 (June 2010) and 1.04 (September 2011) as well as all earlier versions, are now considered obsolete.

InChI Software v. 1.05 includes several significant additions to previous versions.

Large molecules (up to 32767 atoms) are now supported, in an experimental mode. Note that InChIs produced have a prefix ‘InChI=1B’ indicating beta status of these identifiers. Analogously, flag character ‘B’ is used in InChIKey instead of ‘S’ (Standard) or ‘N’ (Non-standard).

Also added is an experimental support of simple regular single-strand polymers (more details are given elsewhere; see also v. 1.05 ReleaseNotes). Note that InChI/InChIKey for polymers also carry a ‘B’ mark denoting their beta status.

Large molecules are supported by already known API calls provided that a new option ‘LargeMolecules’ is supplied by the caller.

Generation of InChI for polymers does require use of the new Ex (extended functionality) API functions GetINCHIEx() and others, see below.

Also added is native API support for direct Molfile to InChI conversion through a new function MakeINCHIFromMolfileText(). This function uses the same Molfile parser as inchi-1 executable thus ensuring that any correct caller of the InChI Library procedure will produce the same result as inchi-1.

A whole new set of API calls, IXA functions, is included. IXA stands for *Extended InChI API*. In particular, it contains new API procedures including low-level functions to deal with atoms, bonds, etc., see dedicated section below in this document.

Finally, the InChI Library is now significantly modified internally to support safe multi-threading execution, both under Windows and Linux.

By default, InChI Software v. 1.06 generates standard InChI. In particular, the standard identifier is generated when the software is used without any specified options. If some options are specified, and at least one of them qualifies as related to non-standard InChI, the software produces non-standard InChI/InChIKey. However, for compatibility with the previous v. 1.02-standard (2009) release, API calls which deal only with standard InChI – for example, GetStdINCHI() - are retained (technically, they provide a pre-customized interface to general-purpose API functions).

Below is a brief description of InChI/InChIKey API functions (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 (see, however, the notes below on newly introduced “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.

##### 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) | | |
| LargeMolecules | *Experimental*  Allows input of molecules up to 32767 atoms  Produces ‘InChI=1B’ indicating beta status of resulting identifiers | Input is limited to not more than 1024 atoms |
| 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 |
| 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 | Set time-out per structure in milliseconds; W0 means unlimited | The default value is unlimited |
| NoWarnings | Suppress all warning messages(default: show) |  |
| OutputSDF | Output SDfile instead of InChI |  |
| WarnOnEmptyStructure | Warn and produce empty InChI for empty structure |  |
| SaveOpt | Save custom InChI creation options (non-standard InChI) |  |

##### 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 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 (standard; if no option supplied) |
|  | |  |
|  | | |
| LooseTSACheck | TODO: describe this | Disabled |
| Polymers | Experimental support of simple polymers | Disabled |
| Polymers105 | Experimental support of simple polymers in older v. 1.05 way | Disabled |
| FoldCRU | In polymer treatment, try to fold constitutional repeating units which themselves contain repeats | Disabled |
| NPZz | Allow non-polymer Zz atoms (pseudoelement) | 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).

#### 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

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.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 bymeans 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 theIXA. Most functions in the IXA require the Handle for an IXA Status Object to be passed as a parameter; anyerror 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.

#### XA\_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 unit (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 Informati**on** 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.

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.