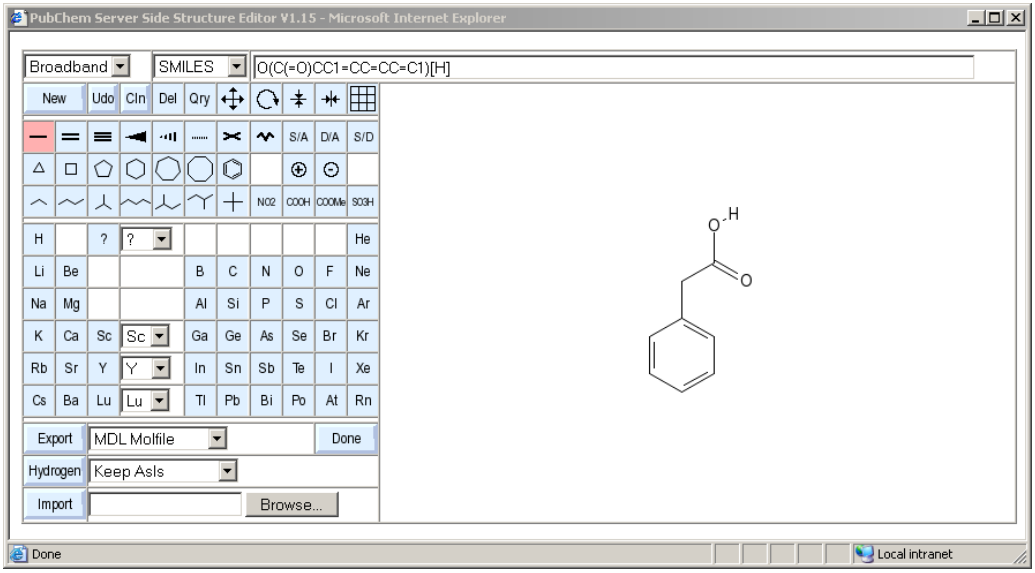
The PubChem Structure Sketcher

Manual for Version 1.20

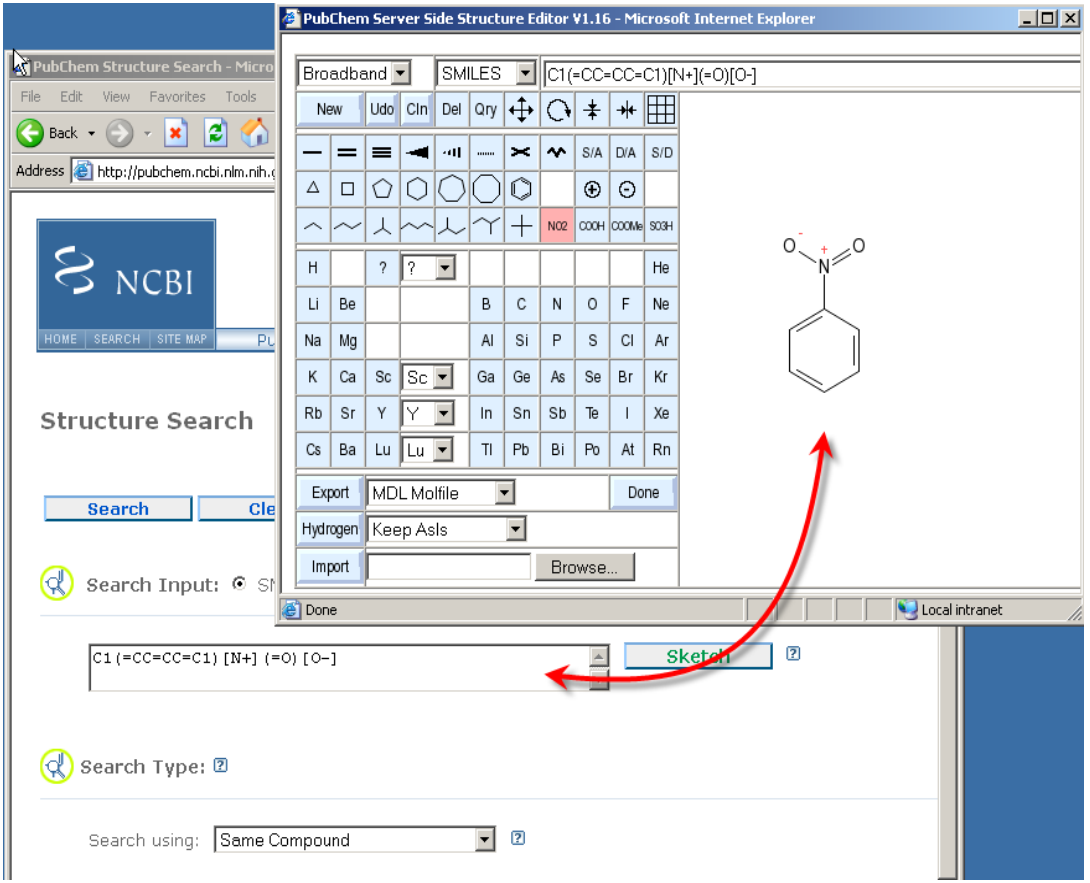


by Wolf-D. Ihlenfeldt

Last update 2007-6-22

CHAPTER 1

PubChem Sketcher User Manual



In order to allow input structures for queries, the PubChem Assay and Structure database uses its own unique Web-based sketcher tool. This document describes its features and operation.

System Requirements

The sketcher was designed to be as Browser- and platform-independent as possible and should work on any recent Web browser on MS Windows, OS X, or Unix/Linux. It does not require the presence of a Java execution environment, nor does it use a plug-in requiring installation and installation privileges. It does not store any data on your computer, or changes the browser configuration, either.

The program works by streaming images to your browser, and capturing mouse events on these images. In order to be able to do that, it requires:

1. A JavaScript interpreter, which does not even need to support the latest Web 2.0 features. Please allow JavaScript interpretation for this tool - without JavaScript support this application will not work at all.
2. A browser capable of displaying PNG and/or GIF images. Some error status is reported via animated GIFs. In order to be able to see error flashes, do not disable image annotation.
3. A reasonably speedy Internet connection. The tool will work via dial-up, but we recommend a faster connection. The total bandwidth consumed by this application is on the order of an Internet radio station and significantly less than any video streaming site.

First Steps

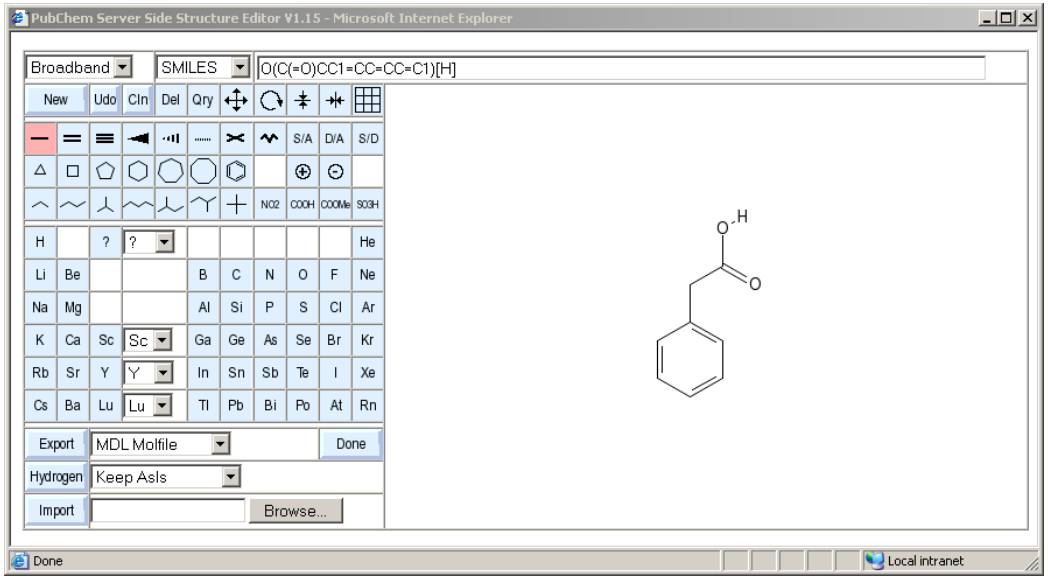
The sketcher is usually started from within a larger form, such as **http://pubchem.ncbi.nlm.gov/search**, is linked to an input form field in that form, and invoked by means of a button labelled *Sketch* right next to it. When this button is clicked, a new window is opened. Depending on the contents of the source field, the sketcher window may be pre-loaded with a structure, for example by decoding a SMILES or InChI string, retrieving a structure via its CID from the PubChem database, or by getting a structure from some other source

Example:

Open page **http://pubchem.ncbi.nlm/gov/search** in your browser. Enter “999” (without the quotes) into the field “Search SMILES...” to specify a CID. Press the *Sketch* button. A separate sketcher window will open that will look similar to the image below.

In case the window does not open, verify that you have JavaScript enabled in your browser. If a popup blocker should interfere, use its override mechanism (such as using ctrl-click) to temporarily allow the

opening of additional windows or, preferably, selectively disable the blocker for the **nlm.nih.gov** domain. The exact methods to achieve this are dependent on the type of blocker you may be using.



Sketcher Operation

Editor Window Layout

A newly opened sketcher window is divided into three distinct zones. To the right the actual sketching area is located. To the left, a number of buttons and other user interface elements control the editor mode of operation. Above the drawing area, there is a text field which displays the current structure in various encodings useful for cutting and pasting. It can also be used to input structure information into the sketcher from structure encodings on the clipboard, or even by typing in structure codes by advanced users.

Button Area

The buttons on the left are used to select the various operations of the sketcher. There are two types of blue buttons. Those without a raised border (element symbols, various bond types, etc.) are mode selectors. When you click them, they are highlighted, and remain highlighted until a different mode is selected, or they

are reset during by the program as a result of a user action. Only one of these mode buttons is active at any time.

The other type of button has a raised profile. These buttons perform an operation immediately when they are clicked. They do not change the overall operating mode of the sketcher. Some of these operational buttons are associated with auxiliary control elements, such as option menus or file upload buttins.

Mouse Use

All operations can be performed with a single-button mouse. If you have more than one button, the left mouse button is used for standard drawing and selection operations. The right mouse button can be used for quick deletions. More about this can be found in the paragraph about deleting objects.

Error Reporting

In case some error condition was encountered while drawing which prevented an operation from performing at least part of its intended work, the drawing area or a part of it will briefly flash orange. If the location of the problem, such as an atom with a valence violation, could be identified, the offending object is flashed as an orange, localized box. If the problem was global in nature, the full background of the drawing area briefly turns orange.

The error flash is generated by sending a specially crafted animated GIF image to your browser. In order to see the flashes, you need to allow image animations in your browser. If you disallow them, no harm is done except that you miss the visual feedback. The software intentionally does not use audio cues.

Bandwidth Control

The choice menu in the upper left corner can be used to lower the bandwidth requirements of the sketcher. This can be useful in case it needs to be accessed via dial-up, or congested connections.

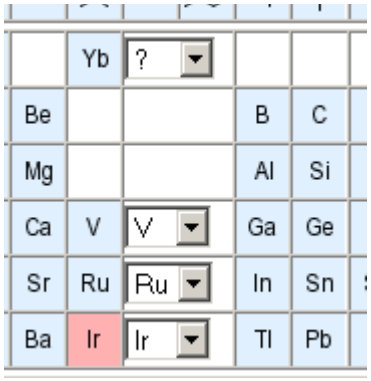
If the choice menu is set to *Dialup*, a few changes in the processing of user input will reduce the amount of data transmitted and make the application more responsive over channels with limited capacity:

- Reduction of the number of mouse tracking events sent when moving the mouse, resulting in fewer image and structure data updates.
- Higher required distance minima for sending update and catchup events in periods of mouse inactivity.

- The drawings will not use anti-aliased fonts and lines, cutting the size of drawing area images in half and thus reducing the amount of data transferred.

Element Buttons

The middle section of the button area is filled by element buttons that are roughly arranged like the periodic table of elements. Clicking one of these buttons will switch the editor to the element mode. When one of these buttons is active, the following operations are supported in the canvas area to the right:



- Clicking in an empty location

  This will add a single atom of the selected type at that location.
- Clicking on an existing atom

  This will change the existing atom to the selected element. If the old atom has bonds, and the number of bonds would result in a gross valence violation, some bonds will be automatically removed.
- Starting on an existing atom, and drag the mouse with left button pressed to a new location

  This will add a new atom at the end position, and attempt to make a single bond to the atom where the operation started.

The button set only displays a single column of buttons for minor group elements. The element any one of these button represents can be changed by means of the option menu to the right of each of these buttons.

In a similar fashion, lanthanides and actinides, as well as deuterium, tritium, and a query ‘any’ atom can be obtained from the button above the minor group element buttons.

Bond Drawing

The second row of buttons provides a selection of different bond types. When one of these buttons are active, the sketcher is in bond drawing mode.

- Clicking on an existing bond

  The order or style of the bond will be changed to the selected type if possible without gross valence violations.
- Start at an existing atom and drag the mouse to a different location

A new bond will be created, beginning from the existing atom. If the mouse button is released on another atom, a bond is created or modified between the start and end atoms. If the location of release is not occupied by any atom, a bond to a new carbon atom is created.

- Clicking on an existing atom

  A new bond to a carbon atom will be sprouted from the start atom. If possible, a 120 degree angle will be maintained to existing bonds on the start atom. If that is not possible, the largest gap in the bond sphere will be filled.

- Start at an empty location, drag to mouse to an different empty location

  A bond between two newly entered carbon atoms will be created using the start and end locations at end points of the bond.

- Start at an empty location, end at an existing atom

  A carbon atom is entered at the location where the mouse was first pressed down, and then a bond is created between the new atom and the end atom.

- Click into empty space

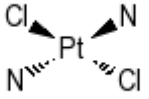
  A horizontal bond between a pair of newly entered carbon atoms will be created.

Special Bond Types

Besides single, double and triple bonds a couple of special bond types are accessible via the bond button set.

- Wedge bonds

  These are used to specify stereochemistry. You may place them anywhere, but they will only specify stereochemistry at atoms which can possibly exhibit tetrahedral stereochemistry, including sulfoxides and similar environments with a free electron pair. A set of four wedges placed on a possible square planar stereocenter is also recognized following the IUPAC recommendations.



- Complex bonds

  The PubChem database uses a special *complex* bond type to encode bonds in complexes which cannot be adequately described by VB bonds. The dotted line bond is used to encode this type of bond. Complex bonds do not participate in electron counting for valence bonds and do not have an inherent bond order. If used as a query bond, they are an “any” bond which will match any database structure bond.

- Crossed bonds

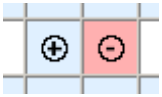
  The crossed bond type is a special type of double bond which is used to indicate that the drawn positions of the bond ligands do not imply defined stereochemistry on a stereogenic double bond.

- Query bonds

  The S/A (single or aromatic), D/A (double or aromatic) and S/D (single or double, but not aromatic) are query bond types which can be used to flexibly define the type of database structure bond which can match this bond if the sketched structure is used for substructure searching. These bonds cannot be used for full-structure lookup.

Atomic Charges

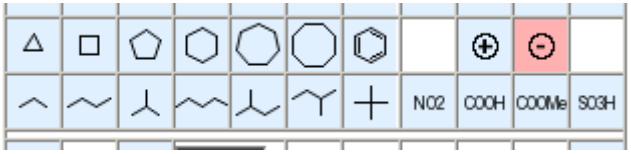
Charges on existing atoms can be specified by selecting the plus or minus charge buttons. If one of these modes are active, a click on an existing atom will increase or decrease the charge by one.



Note that there is currently no support for specifying explicit radicals.

Fragments

Below the bond drawing buttons, two rows of buttons allow the convenient input of larger structural fragments.



The first row of buttons display important basic ring systems. When a button has been activated, its associated drawing mode is used as follows:

- Clink into empty area

  A ring of the selected type is added, with its center at the click position.

- Click onto existing atom

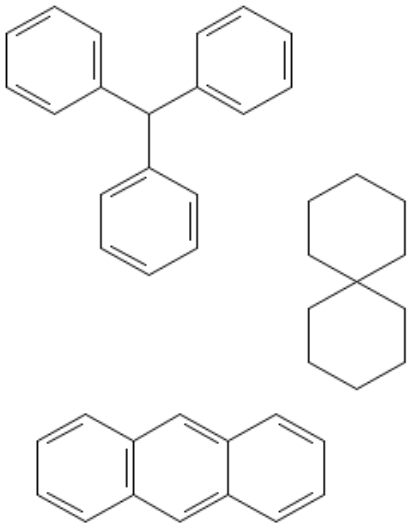
  The selected ring is sprouted from the atom via a single bond, using a 120 degree bond angle where possible.

- Control-click onto existing atom

  The selected ring is sprouted from the atom, incorporating the start atom as first ring atom. If the start atom is already a ring atom, a spiro system is created.

- Click onto existing bond

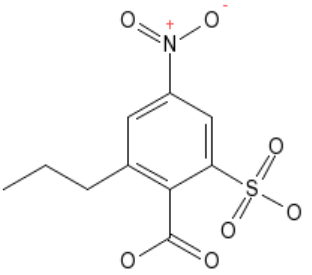
  The ring is annealed to the existing bonds. In case of the phenyl fragment, a smart decision is made about where to put double bonds in the added ring.



In case valence restrictions prevent the full execution of a ring addition, bonds to the source atoms may be omitted.

The second row of buttons displays a couple of important chain fragments and functional groups. These are used in a very similar fashion to the ring fragments, but the spiro or bond addition modes are not supported for them. They can only be added as stand-alone fragments or sprouted from an existing atom.

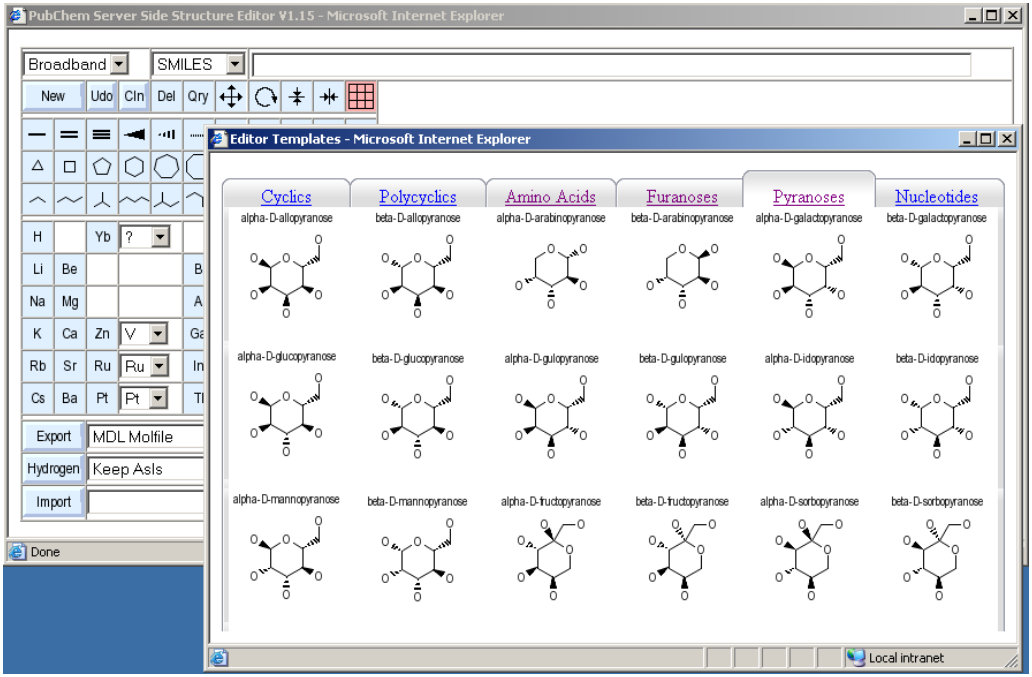
The structure to the right can be built with five mouse clicks into the drawing area, plus five button selections: select the phenyl ring fragment, click anywhere into empty drawing area, select the nitro functional group fragment, click onto the ring atom in the drawing to be substituted, select sulphonic acid group button, again click into drawing area, and repeat twice more for the carboxyl group and the n-propyl group. If desired, a complete set of hydrogen atoms can be added as a final step (see below).



Templates

The fragment button row on the main editor window only shows a small collection of frequently used fragments. A larger template library can be opened by clicking on the grid button in the upper right of

the button section. An auxiliary window with tabs for various types of fragments of biological importance opens.



You can switch between template collections by clicking on the tabs. Individual templates are selected by first clicking onto them, and then into the drawing area where they should be placed. The click position is the center of the fragment placement position. After transferring a fragment into the drawing area, the template window is closed, and the sketcher automatically activates the move mode in order to allow more precise placement of the transferred fragment.

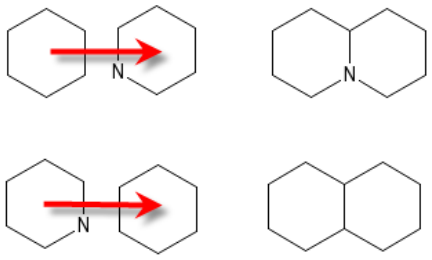
It is currently not possible to automatically link a transferred fragment to existing drawing components at the instant of transfer.

Graphical Manipulations

To the left of the template grid button, the top button row contains four buttons for graphical modifications of the current structure.

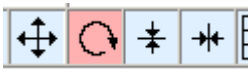
- The *move* mode button (highlighted in the image to the right)

This mode allows you to move structure fragments, atoms or bonds. The position of the initial mouse click determines the object to be moved. If an atom is clicked, only that atom is moved around as long as the mouse key is pressed. All bonds to that atom will adjust. If the clicked object is a bond, both atoms of the bond will be moved in parallel. If neither an atom nor a bond is clicked, but the click point is within the bounding box of a larger fragment on the drawing area (a molecule), the whole fragment is moved.



If at the moment the mouse button is released there are no overlaps between the moved atoms and any other atoms, only the graphical position of the moved objects will have been adjusted. If there are overlaps, an attempt is made to merge the overlapping atoms. Atoms that have not been moved have precedence. In the graphic to the left, if the left ring is moved onto the ring in the middle so that the rightmost two atoms of the moved ring overlap with the leftmost two atoms of the other ring, the results will be as depicted in the right column. If valence restrictions prevent some bonds from being formed, they will be omitted.

- The *rotate* mode



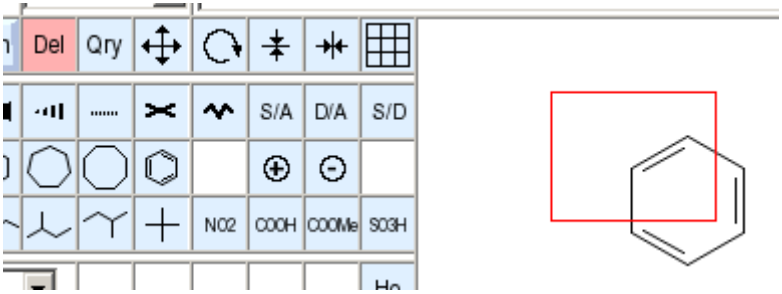
This mode will allow you to rotate fragments on the drawing area by clicking and dragging with the left mouse button. The center of rotation depends on the object at the location where the mouse button was clicked. It can be either an atom, a bond (center of rotation is the center of the bond) or a molecule when the click occurred in the bounding box (the center of rotation is the molecule center). Rotation is currently locked to 30 degree steps. When the rotation is finished, after releasing the mouse button, an atom merging step identical to that in the move mode is performed.

- The *mirror* modes

The mirror buttons allow you to easily generate the mirror image of a fragment, or to change the stereochemistry of a double bond. If a double bond is clicked, the ligands on one side of the bond are flipped so that the compound with opposite cis/trans stereochemistry on that bond results. If the click point is a molecule box, the whole molecule is mirrored along the x or y axis. All tetrahedral stereochemistry and wedge bonds are updated to represent the enantiomer of the mirrored molecule. There is no special action for clicking onto an atom - this is the same as a click into the bounding box.

Deleting Objects

The button marked *Del* is used to enter the object deletion mode. When this mode is active, the following operations are supported:



- Clicking on an atom

  The atom and all the bonds it participates in are deleted.
- Clicking on a bond

  The bond will be deleted, but its atoms remain.
- Clicking into a molecule bounding box, but not onto an atom or bond

  The complete fragment will be deleted.
- Starting in an empty location and dragging

  This is shown in the image above. A red box is displayed which follows the dragged mouse. When the mouse button is released, all objects within the selected area are removed.

If you are using a mouse with more than one button, the right mouse button is a shortcut to deletion opeations. It will always work, without the need to switch into the deletion mode. It supports the quick deletion of atoms, bonds and full fragments. The selection rectangle can only be used in the proper deletion mode.

The quick deletion mode is especially useful when you needed to click into the drawing area, for example in order to assign it the keyboard focus, and by this click inadvertently added a single atom. A quick right click, and the spurious addition is gone.

Undoing, Redoing, and Starting Fresh

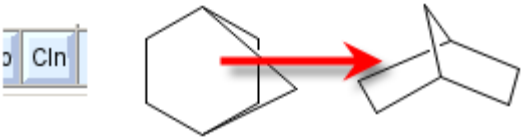
The button marked *Udo* implements a simple undo/redo facility. Only a single operation can be undone. If the button is then clicked again, the undo operation is itself undone, i.e. you end up with the old structure again.

The *New* button deletes the current drawing completely and gives you a blank slate. This operation can also be undone in case the command was executed in error.

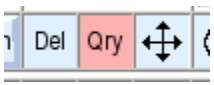
Both buttons perform their operations immediately (as indicated by their raised shape) and are not modes. *Undo* does not change the current sketcher mode, *New* resets it to the single bond drawing tool.

Cleaning up Structures

The button labelled Cln (clean) recomputes the structure layout without changing other aspects of the structure. The image to the right shows a sample molecule before and after cleanup. In case a cleanup should not yield an improved structure layout, it can be undone.

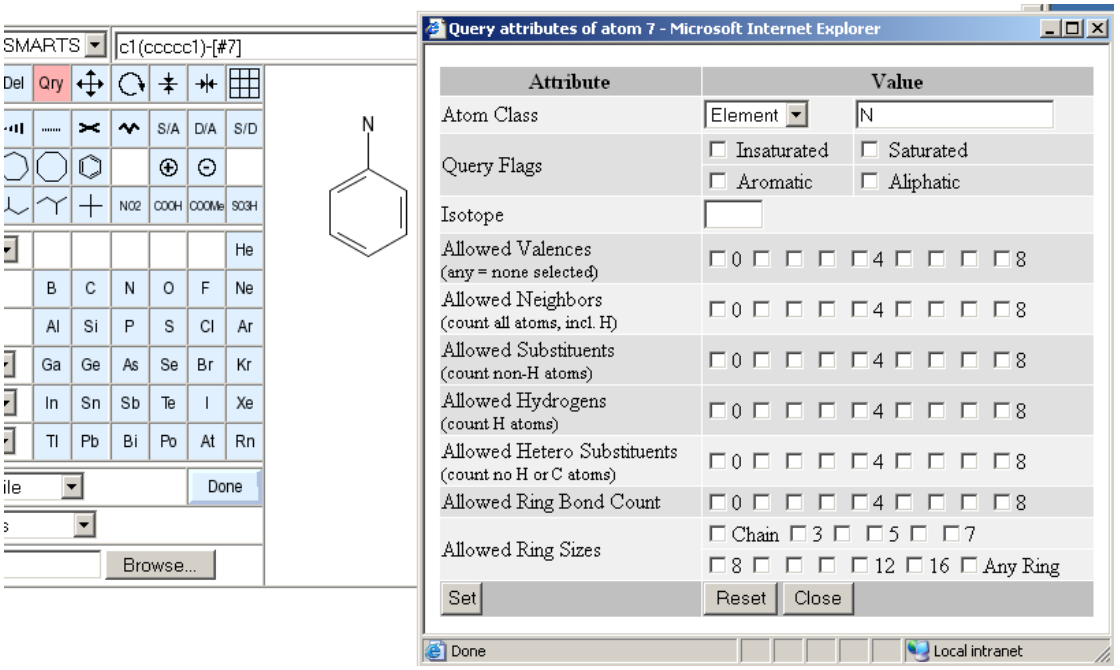


Setting Query Attributes on Atoms



The sketcher supports the setting and deletion of a limited set of query attributes on atoms and bonds. In order to activate the query attribute mode, click the *Qry*

button. In this mode, the left mouse button can be used to click onto atoms or bonds. Depending on the type of object clicked, different query attribute windows will open.



The image above shows the atom attribute window. The following attributes can be set or reset:

- Atom class

  There are four different classes and three predefined element sets accessible in this row. By default, atoms in the sketch correspond to a specific type of physical atom with a defined element. If this atom class is selected, the element may be changed by entering a new element symbol in the text filed to the right. This is equivalent to selecting an element button and clicking onto the atom which should be changed. The other atom classes are *any*, *list* and *negative* list. An *any* atom will match any atom in database structures when used in substructure queries. The text field is ignored for this atom type. A *list* is a set of alternative elements for this atom. You need to specify a whitespace or comma-separated list of acceptable element symbols in the input field to the right. The *negative list* is the same as a normal *list*, except that all elements except the ones specified in the text field will match. Finally, the predefined element sets *hetero*, *halogen* and *metal* are shortcuts for popular element lists. These shortcuts also ignore the text field. Atom classes other than the simple element atom cannot be used for hashcode-based full structure searches.

- Query flags

This set of four check boxes allows you to request saturation or insaturation, and explicit aliphatic or aromatic character of a matched database structure atom. Saturated/insaturated and aliphatic/aromatic are mutually exclusive. These explicit flags which are applied only to a single atom always override any global match conventions set in a general structure search panel which opened the editor window.

- Isotope

  Here you can specify the nucleon count of an isotope label on that atom. The input is a single integer. This information will automatically be used for substructure searches and also, if an appropriate hashcode is used, for hashcode-based full-structure queries.

- Allowed valences

  This is a set of checkboxes where you can set allowed valence states of the atom when matched to a database structure. If no explicit valences are selected, the atom can be of any valence. Note that non-VB bonds (complex bonds, ionic bonds, etc.) in database structures have a zero valence count contribution.

- Allowed neighbors

  This checkbox set works the same way as the valence row above, except that the bonded neighbors are simply counted, and bond orders and bond types are ignored.

- Allowed substituents

  Essentially the same as the neighbor count, except that hydrogen is not counted in the bonded neighbors.

- Allowed hydrogens

  This constraint is complementary to the substituent and neighbor counts.

- Allowed hetero substituents

  Essentially the same as the neighbor count, except that neither hydrogen nor carbon bonded neighbors are counted.

- Allowed ring bond count

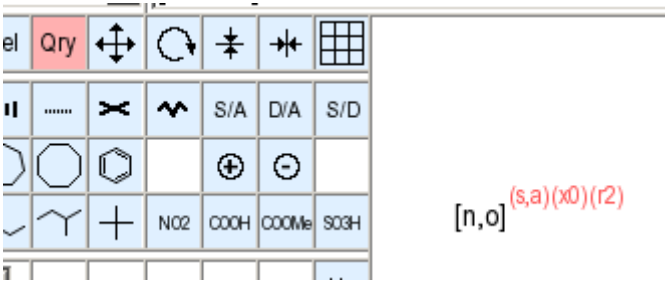
  The number of ring bonds the atom may participate in. Note that both VB and complex bonds can be ring bonds, but ionic bonds and other exotic types are excluded in the ring bond detection and are never considered part of a ring.

- Allowed ring sizes

  If you select one or more of these checkboxes, the atom can only match to database structure atoms which are a member of a ring of at least one of the selected sizes, or specifically a chain atom. The ring set used to determine this property is dependent on the back-end structure database system. For PubChem, it is an extended set of smallest rings in which all 3-atom sequences of bonded ring atoms are part of at least one ring. This ring set is more symmetrical than the classical SSSR.

Display of Atom Query Attributes

Any atom query attributes which are set in the atom query panel, or which are already present when a structure is pre-loaded or imported, are reflected in the drawing on the canvas. Atoms with query attributes are



drawn with extended atom symbols and/or read attribute annotations. These special element symbols are used for extended symbols:

- An atom list (lowercase for aromatic atoms) [Cl,Br,I]
- A negative atom list [!S,!Te]
- An *any* atom ?
- A metal atom M
- A halogen atom X
- A hetero atom XX
- and these characters for additional attributes plotted as red markers to the upper right of the atom symbol:
- Charges in normal style as “++” or “3-”
- Isotopes [nucleon count (integer)]
- Saturated atom s
- Insaturated atom u
- Aromatic atom a
- Aliphatic atom A
- Valence V (as in “V3,5”)
- Neighbors X (as in “X4”)
- Substituents D (as in “D3”)
- Hydrogens H (as in “H2”)
- Hetero substituents x (as in “x1-2”)
- Ring bonds r (as in “r3” or “r-1”)

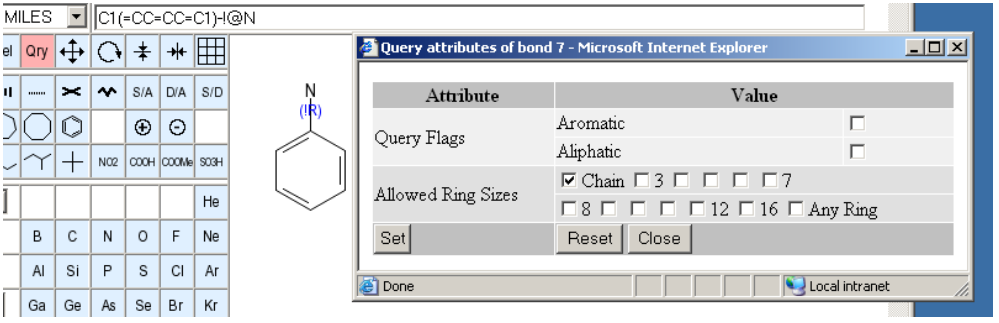
- Allowed Ring sizes R (as in “R3”, “!R” for chain)

For query attributes with alternative possible counts, an attempt will be make to contract the displayed set as much as possible using closed (“1-2”) and open (“-1” or “4-”) ranges and lists of alternative single values (“3,5”).

The structure handling library used to implement this applications supports many more query attributes than those accessible via the atom attribute panel. In case these were imported by reading a file with a query specification, other attributes may be displayed in addition to those described in this section. As long as they are not overwritten by explicit setting of new attributes on affected atoms, they will, if possible in the structure data transfer format, be preserved in structures submitted via the editor.

Setting Bond Query Attributes

If the editor is in query attribute mode, and you click onto a bond, this query attribute window opens:



Its mode of operation is very similar to the corresponding atom panel, but there are different and fewer attributes which can be set and changed.

- Query flags

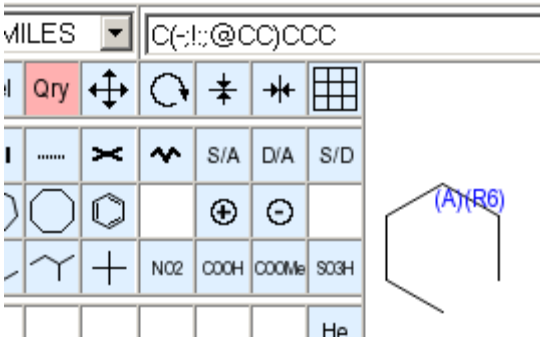
  You can request a bond to be matched only to aromatic or aliphatic database structure bonds.

- Allowed ring Sizes

  If you select one or more of these checkboxes, the bond can only match to database structure bonds which are a part of a ring of at least one of the selected sizes, or specifically a chain bond. The ring set used to determine this property is dependent on the back-end structure database system. For PubChem, it is an extended set of smallest rings in which all 3-atom sequences of bonded ring atoms are part of at least one ring. This ring set is more symmetrical than the classical SSSR.

Display of Bond Query Attributes

As for atoms, query attributes of bonds are displayed on the drawing area. The annotation color for bond attributes is dark blue, and attribute annotations are drawn over the center of the bond.



These characters are used to display bond query attributes:

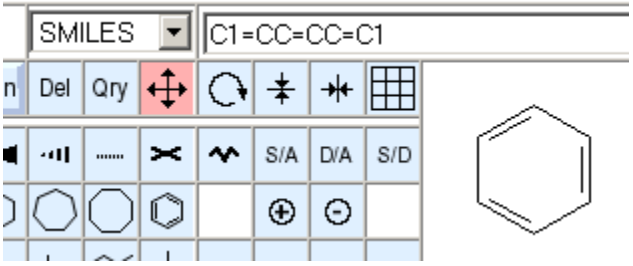
- Aliphatic bond — A
- Aromatic bond — a
- Ring sizes — R (as in R6, R-5)

For query attributes with alternative possible counts, an attempt will be make to contract the displayed set as much as possible using closed (“1-2”) and open (“-1” or “4-”) ranges and lists of alternative single values (“3,5”).

The structure handling library used to implement this applications supports many more query attributes than those accessible via the bond attribute panel. In case these were imported by reading a file with a query specification, other attributes may be displayed in addition to those described in this section. As long as they are not overwritten by explicit setting of new attributes on affected bonds, they will, if possible in the structure data transfer format, be preserved in structures submitted via the editor.

The Structure Data Line

Above the drawing area, a text field displays continuously updated information about the currently edited structure. The type of data displayed can be changed by the choice menu to the left of the text field.

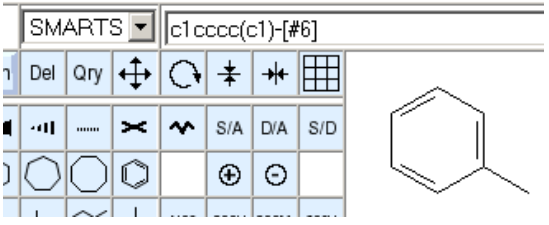


The following choices are available:

- SMILES

  The text line shows a SMILES encoding of the edited structure, assuming that hydrogens are implicitly added. The encoding of aromatic systems is in Kekulé form for maximum easy of decoding.

- SMARTS

  The text line displays a SMARTS encoding. The atom aromaticity atom attribute (lowercase element symbols) is automatically set for identified aromatic systems in the drawing. Atoms for which the aromaticity status cannot be determined are encoded as element numbers ([#6] for carbon) to avoid implicit assumptions about their aromaticity when decoding. Aromatic bonds are encoded as implicit bonds, aliphatic single bonds use explicit single bond encoding ([#6]-[#6]).

- InChI

  The line displays an InChI encoding of the structure. InChI is a new IUPAC standard for compact, unique representation of chemical structures. A full set of implicit hydrogen is assumed in the encoding. All hydrogen atoms are encoded with fixed positions so that the structure decodes to exactly the same tautomer as drawn.

- Formula

  The line displays the molecular formula and molecular weight. A full complement of hydrogen atoms is implicitly assumed.

- SLN

  The line content is a Sybyl Line Notation encoding of the current structure.

The text in the data field can be conveniently be copied and pasted with normal text highlight and clipboard operations. Thus, the sketcher can be a convenient input tool even for sites which do not possess a direct data update link to the main query form but do allow their structure input data to be encoded in SMILES, SMARTS or SLN.

In the SMILES, SMARTS and SLN representations, atom and bond query attributes are encoded as far as technically possible. Not all supported query attributes can be expressed in all of these line notation formats. The sketcher SMILES encoding uses the following custom extensions:

- Complex bonds | as special bond character
- Allowed hetero substituents *x* as query atom attribute

On some sites, the installer of the sketcher application may have opted to automatically and continuously copy the information from the data line to the clipboard after each data change. This only works with the Internet Explorer Web browser and removes the need to manually copy the text content. The PubChem site does not employ this optional feature.

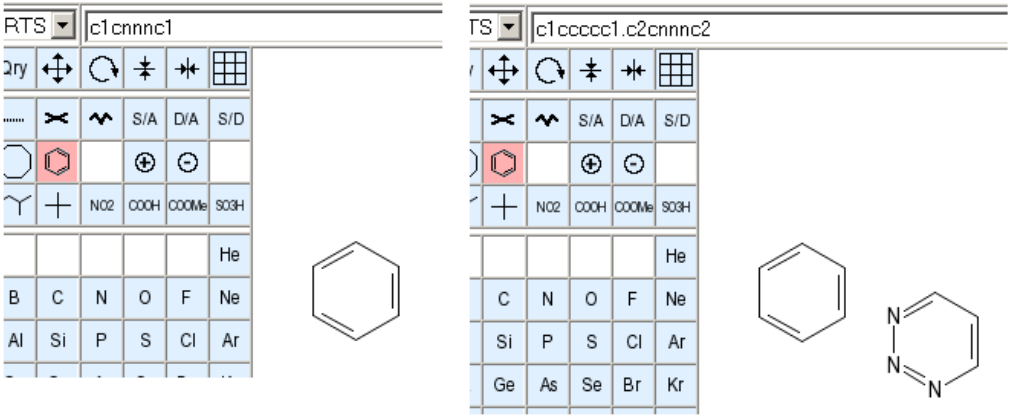
Structure Input via the Structure Data Line

The structure data line serves not only as an information display. It is possible to enter structure codes in this field and import structure data into the drawing area.

Structure import is performed by clearing the line, and then editing or pasting a structure code into this field and finally pressing the return key. The setting of the display choice menu to the left of the field has no influence on the operation.

If the decoding of the structure data succeeds, the encoded content will be merged into the existing drawing as a new additional fragment. If it is intended to replace the current contents, you need to clear the drawing

area first by pressing the *New* button. Fragments are imported without implicit hydrogens and are placed automatically.



The images above display the drawing area before and after the SMILES string *c1cnnnc1* was imported. Note that the implicit aromatic system in the input string was automatically resolved to a proper Kekulé form.

The complete set of supported structure data string formats depends on the back-end installation. The following will always work:

- SMILES/SMARTS strings
- InChI strings
- CACTVS toolkit hex-encoded compressed binary blobs

Additional formats which may be supported in specific installations:

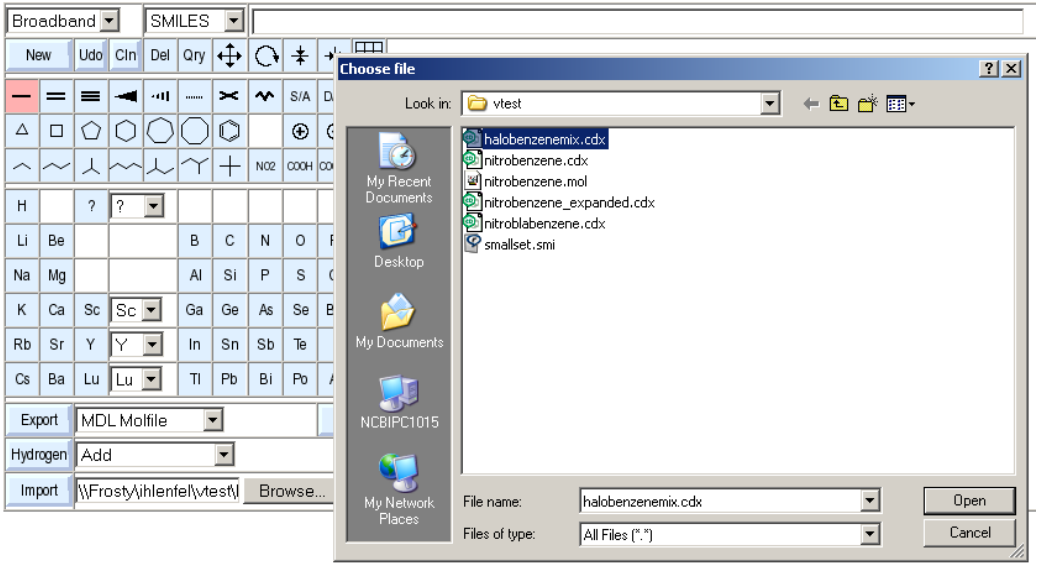
- Hex-encoded SMILES
- CACTVS toolkit Minimol strings
- SLN (Sybyl Line Notation) strings
- Hex-encoded compressed Molfile strings (as used by MDL Chime)
- JME strings (as used by the MolInspiration Java applet sketcher)

Structure Import via Keyboard Paste

The supported strings that can be imported via the data line may also be directly pasted from the clipboard into the drawing area by means of the standard ctrl-V keyboard shortcut, provided that the drawing area has keyboard focus. This method has the additional advantage that the location of the mouse at the moment of the keypress determines the location of the center of the newly added fragment. Depending on the browser and client operating system, the drawing area may not automatically have keyboard focus when you move the mouse into it. To make sure that it has, click the mouse once. In case this leads to the addition of an unwanted atom or a fragment, use the right mouse button to delete these.

Structure Import via File Upload

The final method for loading existing structure data into the sketcher is by means of file upload.



In order to do this, select an existing structure file via the *Browse..* button and then press the *Import* button. The file is then read and added to the existing content. In case you want to guarantee that the imported file is the only sketcher content, press the *New* button before the import.

The import function only reads the first record of multi-record files. So in case you attempt to upload an SD-file, only the first record will show.

The hydrogen status of the imported structure will be adjusted at upload time depending on the setting of the *Hydrogen* option menu. These are its possible values:

- Add

  A standard set of hydrogens is added to all open valences.
- Add Special

  Hydrogen is added to all hetero atoms and carbon atoms where it is needed to make the encoding unambiguous, i.e. at stereo centers and stereo bonds, as well as to carbon atoms which traditionally are drawn with explicit hydrogens (aldehydes, C triple bond terminals, etc.).
- Keep AsIs

  The hydrogen status is kept as it was in the upload file.
- Strip Non-Special

  Hydrogen is remove from carbon atoms, except where it is needed to determine stereochemistry, or where it is traditionally drawn (aldehydes, C triple bond terminals, etc.)
- Strip All

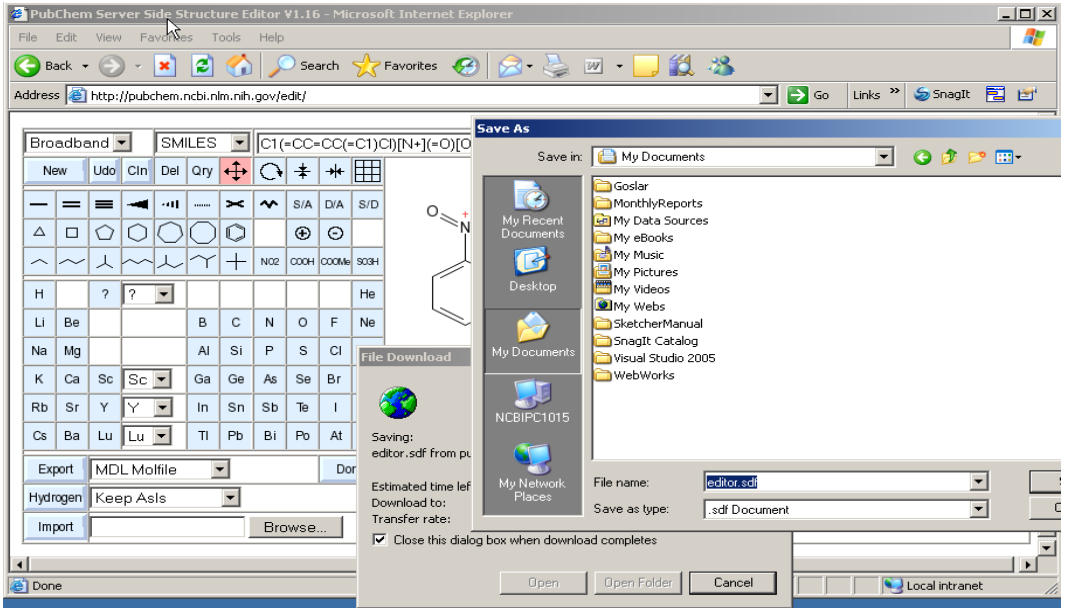
  All hydrogen atoms are removed from input.

The exact set of supported file formats for sketcher import is dependent on the installation. In general, almost any frequently used chemical structure file format will be understood (more than 3 dozen file formats and format variants), but specific installations may choose to reduce the set of file formats they are willing to support.

Structure Export

The sketcher can export the currently edited structure in various formats. This capability makes it a convenient tool for the input of data sets of limited to be used locally, or even file format conversion.

The current structure is exported by clicking on the button labelled Export. A file selector box will open and let you specify the name and location of the file downloaded from the sketcher server. The default name of



the download file is *editor.xxx*, where *xxx* is a suitable default suffix for the selected file format.

The desired format of the file can be selected, before clicking the *Export* button, from the menu to the right of that button. The exact list of formats which can be used for export is dependent on the installation. The default set of formats includes structure exchange formats (MDL Molfile, SMD, SMILES, SMARTS, CML), structure editor formats for further clean-up of the drawings for publications etc. (ChemDraw CDX and CDXML, Isis/Draw SKC) and image downloads (GIF, PNG, SVG, EPS). In default installations, mandatory 3D formats (PDB, MOL2, XYZ, etc.) are not available because the sketcher back-end is not installed with a version with an integrated 3D coordinate generator.

If the export format supports it, query attributes will be encoded in the export data as far as technically possible. However, not all formats can express all supported attributes. For example, MDL ISIS query files (a special subclass of the MDL Molfile) cannot encode the atom aromaticity query attribute which is popular in SMARTS. On the other hand, standard SMARTS has no ring bond count attribute which is available in ISIS. Query attributes which cannot be expressed in the selected export format will be silently ignored.

The native CACTVS binary export format plays a special role in the context of structure export and restoration. Because it is the underlying format of the sketcher back-end engine, It is the only format which is guaranteed to be lossless. Only with this format, the original structure can be restored under all circumstances

with all attributes and other features exactly as they were when the structure was exported. Nevertheless, for standard structures without special features, a plain MDL Molfile will work well, too.

Hydrogen Manipulation

Besides its role for file import and export of structure data, the *Hydrogen* option menu can also be used for direct manipulation of the currently edited structure.

Simply set the menu to the desired operation, and press the *Hydrogen* button to the left of the menu. The hydrogen status of the drawn structure will be immediately adjusted.

Keyboard Shortcuts

Many of the sketcher modes can be controlled by keyboard shortcuts. This feature allows advanced users to keep the mouse in the drawing area without moving it left and right to switch buttons.

These are the shortcuts:

- Ctrl-B Beautify structure. Does not work on all platforms.
- Ctrl-C Copy structure as SMILES onto the clipboard. Does not work on all browsers.
- Ctrl-N New drawing. Do not use on MS Windows - it clones the sketcher window instead!
- Ctrl-V Paste structure fragment at current position. See separate paragraph on keyboard pasting.
- Ctrl-X Copy structure as SMILES to clipboard, and erase current drawing.
- Ctrl-Y Undo last operation.
- Ctrl-Z Undo last operation.
- A Select *Al* as current element
- b Select *B* as current element
- B Select *Br* as current element
- c Select *C* as current element
- C Select *Cl* as current element
- d Select *D* (2H) as current element
- f Select *F* as current element
- F Select *Fe* as current element
- G Select *Ge* as current element
- h Select *H* as current element
- H Select *Hg* as current element

- i Select *I* as current element
- I Select *In* as current element
- k Select *K* as current element
- K Select *Kr* as current element
- L Select *Li* as current element
- m Enter *move* mode
- M Select *Mg* as current element
- n Select *N* as current element
- N Select *Na* as current element
- o Select *O* as current element
- O Select *Os* as current element
- p Select *P* as current element
- P Select *Pd* as current element
- q Enter *query* mode
- r Enter *rotate* mode
- R Select *Ru* as current element
- s Select *S* as current element
- S Select *Si* as current element
- t Select T (3H) as current element
- T Select *Te* as current element
- u Select *U* as current element
- v Select *V* as current element
- w Select *W* as current element
- x Enter *delete* mode
- X Select *Xe* as current element
- Y Select *Yb* as current element
- Z Select *Zn* as current element
- > Enter *down* bond drawing mode
- < Enter *up* bond drawing mode
- + Enter *plus* charge change mode
- - Enter *single* bond drawing mode

- = Enter *double* bond drawing mode
- # Enter *triple* bond drawing mode
- / Enter *either* (crossed double bond) mode
- | Enter *any* query bond mode
- ? Select *any* query atom as current element
- * Select *any* query atom as current element
- 0 Activate *phenyl* ring template
- 1 Enter *single* bond drawing mode
- 2 Enter *double* bond drawing mode
- 3 Enter *triple* bond drawing mode
- 4 Activate 4-membered ring template
- 5 Activate 5-membered ring template
- 6 Activate 6-membered ring template
- 7 Activate 7-membered ring template
- 8 Activate 8-membered ring template

The mnemonic for element shortcuts is that lowercase letters select the element where the single-letter symbol corresponds to the pressed key. Uppercase letters select the most important element with a two-letter symbol which starts with the pressed key.

There is no 9-membered ring template which could be associated with key 9.

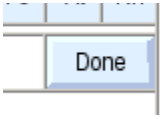
Data Transfer to Caller Forms

A question which has been asked more than once concerns the problem of how to transfer the edited structure data from the sketcher to a linked form which opened the sketcher window.

The answer is simple: There is absolutely nothing a user needs to do to achieve this. Data transfer is automatic, and dynamic. Every structure change is immediately reported to the caller form. There is no button which needs to be clicked in order to transmit the currentl sketch.

Quitting the Sketcher

As described above, there is no need for any user action to transmit the structure data to an originating form for further processing. The sketcher can thus be quit at any time, without fear of data loss, simply by closing its window by means of the standard mechanisms of the client platform, such as clicking on the cross-shaped close icon on the upper right of the windows on MS Windows.



Nevertheless, since in our experience many users appear to be more comfortable if they can hit a dedicated, clearly labelled button to finish the arduous task of inputting an important query structure, we have added a big *Done* button to the sketcher button set. It is located to the right of the *Export* controls.

This prominent button simply closes the sketcher window and does nothing else.

CHAPTER 2

Installation of a Sketcher Web Service

Theory of Operation

The sketcher may be run in various modes. The exact mode of operation must be configured at set-up and cannot be changed without terminating the sketcher server(s) and doing a re-configuration.

All interactive functionality of the sketcher is controlled by a single server program. This server may be run either as a stand-alone process or daemon, or as a standard FCGI application controlled by the main Web server.

Operation as Stand-Alone Process

Operation as a stand-alone daemon or process is primarily useful for debugging purposes. Since in this mode the software can write trace output to standard output, and it can run under any developer account, and can easily be killed without the need to do this as root or with the permissions of the Web server process, customizing the application is much more convenient in this mode. The major disadvantage of running the sketcher as a separate process is that it needs to use a different port for communication with the client than the Web server transmitting the auxiliary HTML pages. This leads to cross-site JavaScript execution problems even if the main Web server and the sketcher process are executing on the same computer. The reason is that, according to a strict interpretation of JavaScript permission rules, JavaScript access across windows is disallowed when the window contents were loaded via different ports, even if the contents came from the same host. There is disagreement among browser developers whether different ports of origin but same host of origin should lead to blocking of cross-window scripting. MS Internet Explorer will still allow this, while most other browsers block JavaScript access in this environment. In practice this means that development and debugging of new Sketcher features will require the use of Internet Explorer. Verification of operation on other browsers can only be performed once the updated version has been re-configured to run as an FCGI.

Operation as FCGI Process

FCGI processes are controlled by a Web server, and both input and output are routed through the Web server so that all data transfer appears to involve only the Web server and its port.Therefore, there are no cross-window scripting problems.

In order to run the sketcher as an FCGI, the main Web server must have been configured to support FCGI. For Apache, this means that standard module *mod_fcgi.so* needs to be present and loaded. In addition, the sketcher installation directory must be set up in the Web server configuration to allow CGI execution. Currently, all sketcher components need to be served from a common installation directory. There is no provision for a split into the FCGI server component and supporting static files.

The size of the FCGI program is not trivial, and initialization after start-up takes maybe half a second. Since sketching events need to be processed much faster than this, running the application as a standard CGI which is started anew for every processing request and then terminates is not a viable option.

Storage of Structure Information

The sketcher obviously needs to have a memory on what a specific user is currently drawing. The client JavaScript code only reports drawing events, but no information about the current structure. This information must be stored somewhere between events.

The simplest method to store structure data is to keep it in the memory of the sketcher FCGI program. This mode is simple to use and probably the most useful for the majority of application scenarios. Since clients transmit an unique session ID, multiple users can use the same sketcher process (seemingly) simultaneously. The sketcher server can keep multiple structures that are edited in different sessions byt the same or different users in memory. By correlating the session ID with its internal storage manager, the correct structure among the current memory set is selected, edited, and the results sent back to the client. There is no hard-coded limit on the number of parallel editing sessions which can be maintained. On an average Web server host, a couple of dozen parallel editing sessions are easily feasible. As long as no events are received from a client there is only a storage requirement of a few kilobytes of memory, but no further load on the server because there is no computation and traffic associated with that session.

In addition to internal memory storage, the sketcher application supports storage of state information in external storage managers. The default application script supports the NCBI NetCache network data caching system, and the PubChem QueueManager queueing system. If the use of any of those systems is enabled in the sketcher configuration, structure state is swapped out to the external storage manager immediately after every operation, and all internally memorized state information is deleted. When a client event is received, the session ID information is used to first re-load the current structure data

before the actual processing of the event begins. The overhead for retrieving and storing structure state is not big, just a few milliseconds.

The big advantage to external storage of structure state information is that in that case multiple sketcher processes can operate in parallel, on one or more physical servers. This way, the software becomes both scalable and robust against failures of single servers. At PubChem, the site *http://pubchem.ncbi.nlm.nih.gov/edit/* is actually served by two physical computers, named *pubchem3* and *pubchem4*. Access to these two systems is dynamically load-balanced, and both servers run their own multiple sketcher FCGI instances. During the course of an editing session, the session is typically bounced between hosts multiple times, but in a completely transparent fashion and without loosing any sketcher content because every sketcher server process knows where to obtain the most current structure information to proceed with the drawing event to be processed next. The storage managers are typically itself mirrored for performance and reliability.

This manual assumes that any external storage manager which is going to be used is already installed and configured properly.

In case in-memory state storage is used, the Web server must be prevented from running multiple instances of the sketcher FCGI by appropriate main Web server configuration options. Since state is stored only in a single process and no inter-process communication is supported, a switch to a second FCGI process by the Web server will lead to a loss of the current structure information. Multiple FCGI instances are allowed to be run on a single machine if external state storage is used. This is primarily useful for multi-processor hosts.

Preparing for Installation

To prepare for installation, create an empty temporary staging directory. Download the Sketcher installation package, or obtain it in another way, and unpack it in this staging directory. The standard package format is a simple *.tgz* file, so the commands should be something like

```
mkdir stage
cd stage
gunzip </my/path/edit.tgz|tar xvf -
```

After installation, the contents of the staging directory can be deleted.

Configurable Template Directories

After unpacking a sketcher distribution in an installation directory, there are two subdirectories which contain data which may be customized.

The *img* directory contains small images used as buttons on the sketcher interface pages. Examples are element symbols and bond type icons. These images were not hand-drawn but can be regenerated programmatically by running the *mkicons.tcl* script with a sketcher FCGI interpreter, as in

```
csweb -f mkicons.tcl
```

The *mkicons.tcl* script may be edited to use different fonts and backgrounds, etc. Normal installations will probably stick with the images in their standard supplied form.

The second directory named *tpl* contains script code and HTML templates which need to be adapted to set up the service in the current location. Files from this directory are processed by the supplied *Makefile* and stored in modified form in the target directory, overwriting any older processed interface files in that location.

Selecting a Script Interpreter

The standard sketcher package does not contain the script interpreter which runs the application script. For default installations, a suitable interpreter is the *csweb* executable found in academic and commercial CACTVS toolkit packages.

There are two interpreters commonly used with this application:

- csweb — This is a standard CACTVS Toolkit interpreter, without any NCBI extensions. It will only work with the in-memory storage system for sketcher structure data. It will also not be able to import, for example, PubChem structures via CID reference or perform any other tasks which require interaction with NCBI-specific systems or encodings.
- csweb_nlm_static — This is the default CACTVS Toolkit interpreter used in a variety of PubChem Web services. Its functionality is a superset of the standard *csweb* interpreter. It has support for interacting with the PubChem database, knows how to talk to the *qman* and *netcache* storage manager systems, supports PubChem ASN.1 encoding and decoding of structure data and various other enhancements for operating in the PubChem environment.

To prepare for an installation, either copy the selected interpreter executable into the unpacked installation directory, or make at least sure that it is found in the standard path.

Configuring the Installation

The key to setting up a custom installation is the *Makefile* found in the staging directory. The first lines of the *Makefile* contain variable definitions which are used to process template files and update their internal references etc. so that the whole set-up works in the location of the target directory.

The following variables need to be set:

- TARGETDIR

  The full path name of the target directory for the installation, as seen in the file system. Usually it is a subdirectory of *htdocs*. An attempt will be made to create the directory if it is not yet present.

- INTERPRETER

  The name of the interpreter executable, without a path.

- SERVER

  Allowable values are *standalone* for the user process/daemon version, and *fcgi* for FCGI. A standard non-developer installation will use *fcgi*.

- STANDALONEPORT

  This variable is only used for the standalone version. It defines the port the sketcher process will be operating on. It must not collide with the port of any other network-based service on the system. In addition, using a port below 1024 will probably require special permissions.

- STORAGE

  This is the selected storage system. It can be *memory*, *netcache* or *qman*. The *netcache* and *qman* options require that the respective service has been set up and is running.

- CACHEPORT, CACHEHOST and CACHESERVICE

  These parameters are of relevance only if the *netcache* storage system is used. Please refer to the *netcache* documentation for details.

- HOST

  This is the fully qualified name (computer name and domain) of the server host this application is being installed on, as visible from the outside Internet. In case a virtual host is used which comprises of more than one physical host, use the virtual host name.

- DOMAIN

  This variable defines the document domain the sketcher will be operating in. It can be set either to the same value as the HOST variable, or it can be any more generic Internet domain. For example, if the host is *pubchem.ncbi.nlm.nih.gov,* it may be set to the same value, or *ncbi.nlm.nih.gov*, or *nlm.nih.gov.* This parameter effectively controls from which sites the sketcher functionality can be used from. If the fully qualified host name is used, only pages served from the same host can receive sketcher data in forms and other page elements. If a more generic domain is configured, any host which resides in that domain can

use the sketcher. The problem of using document domains and its impact on Web pages wanting to receive sketcher data is discussed in Chapter 3 of this manual.

- WEBDIR

  The name of the installation directory, without leading or trailing slashes, as seen from the Web server root. For example, if the installation directory was called */www/htdocs/edit* in the local file-system, it may appear has *http://$(HOST)/edit* on the Web. In that case, the Web directory name set here would just be *edit*.

- TRANSFER

  This variable controls which type of data the sketcher will attempt to send to an opener page after every editing operation which changed the structure. The mechanism used to receive sketcher data on form pages etc. is described in Chapter 3 of this manual. The configuration here selects only available data, but does not require that any opening form page actually makes use of it. The parameter is expected to be a space-separated list of the desired transfer formats. The available formats include *smiles* (SMILES or SMARTS string, depending on sketcher window controls), *sln* (Sybyl line notation string), *jme* (JME Java molecule editor string), *inchi* (IUPAC InChI string), *keys* (Netcache session keys, only if Netcache storage manager is used), *sessionkey* (the current session key), *blob* (CACTVS compressed serialized structure blob in base64-encoding), *molfile* (MDL Molfile image) and *formula* (molecular formula of edited structure, with implied hydrogen). CACTVS blobs and Molfile images can have a size of a few kilobytes each, while all other data is roughly 100 bytes per item. Since this data is transferred from the sketcher server to the client browser after every drawing even which modified the structure, transfer of items in these larger data formats can have a notable impact on sketcher usability over slower connections and thus should only be enabled if a format is actually needed by any of the Web applications receiving sketcher data from an installation. Use of lossless CACTVS blobs or MDL Molfile record images has a place, though, because there are, for example, query attributes which can be set in the sketcher and which cannot be transmitted in other formats.

- DEBUG

  Set this value to 1 if you want debug output on the processing of individual sketcher events on standard output. This option can only be used with the *standalone* server mode, because otherwise standard output is needed for data transfer with the main Web server.

- IMGW and IMGH

  This is not the size of the sketcher main drawing area but the size of template depictions in the template panel. You probably do not want to change this except in case you have changed the default templates in file *tpl/tpl.tpl* to a set with significantly different size characteristics.

Performing an Installation

After the *Makefile* in the unpacked staging directory has been edited, perform the following steps:

- In case any sketcher server processes are running that access the target directory, kill them. Killing Web server-controlled FCGI processes may require special permissions.
- Execute the command

  ```
  make install
  ```

  in the staging directory. This will fetch a fresh set of configuration-dependent files form the *tpl* subdirectory, process them, and copy them to the installation directory. Any old files from earlier installs will be overwritten. Some static HMTL files and the script interpreter will also be copied.
- In case the operation mode is *standalone*, an executable called *editsrv* is assembled in the installation directory. It can either be manually started from there, or via some kind of custom */etc/init.d*. init script. In any case, the Web server will not be able to start it automatically when a sketcher page with dynamic content is loaded. To verify proper operation of the sketcher, the server process must be started manually before any sketcher pages are loaded.
- In case the operation mode is *fcgi*, an executable called *editsrv.fcgi* is assembled in the installation directory. A properly configured Web server should start it automatically if a sketcher page with dynamic content is loaded. If this does not happen, consult the error log of the Web server for information about the problem. In case of an earlier start-up failure or crash due to a misconfiguration, many Web servers impose a blackout period of 10 minutes or more before they will attempt any additional automatic restart of an FCGI. In that case, even a corrected set-up may appear to fail for a period of time. If it is not a production system, restarting the main Web server may be an option.
- Verify that the sketcher is working by loading *index.html* into a client Web browser. Use the Web path of the target directory. You can also simply use the target directory URL if *index.html* is the default directory view file as configured in the Web server. Next try a few structure drawing operations. Any failure is likely to become immediately obvious by displaying a broken canvas drawing area image. In case a virtual server with multiple physical servers is used, it is rather unpredictable when any of the physical servers will be contacted. In that case we can only suggest a longer sketching test session - we have seen cases where a switch to a specific physical server with a problem only happened after a few minutes, and until that moment no problems were obvious because the faulty server was never contacted.

Cleanup

After a successful installation, the staging directory can be deleted.

CHAPTER 3

Linking the Sketcher to Web Pages

General Introduction

The Sketcher program was designed to be easy to use in a variety of Web linking scenarios, without the need to customize separate installations for each application, and with minimum impact on the design of Web pages making use of the sketcher.

General Integration Method

Usually, the sketcher is invoked by opening up a separate sketcher window by clicking onto a button. The associated JavaScript fragment looks like this:

```
var editorwin = open("../edit/index.html",
  "editor","height=440,width=880,scrollbars=no,status=yes,location=no,menubar=no,toolbar=no,resizable=yes",true);
```

The default size of the sketcher window is 440 by 880 pixels. This will display well on all standard browsers. For the sake of users with non-standard font size settings, the subwindow should be opened as resizable.

In theory, it is also possible to load the sketcher into a frame or iframe in the context of a larger window. However, the required window size of the sketcher is rather large, and pages which embed the editor thus tend to appear crowded.

Pre-loading Sketcher Content

The editor can be opened with initial content. This is done simply by providing CGI parameters to the opening statement, as in

```
editorwin = open("../edit/index.html?smiles=c1cccnc1",....
```

The following CGI parameters are recognized:

- smiles A SMILES string of a chemical structure. The structure will be decoded without implicit hydrogens. When providing SMILES and SMARTS strings as initialization data, look out for

the # character which needs to be escaped in order to prevent it from being misread as a page location URL component.

- smarts — A SMARTS string of a query structure. The query will be fully decoded, but not all SMARTS query features can be displayed in the sketcher, and not all possible atom and bond query attributes can be edited. As long as atom or bonds with extended SMARTS attributes are not touched, these extra attributes will however be preserved and exported in suitable formats.
- jme — An editor string for the popular JME editor[1] applet.
- sln — A Sybyl line notation string of a query structure. Support for SLN query attributes is not complete.
- minimol — A CACTVS V2 base64-encoded minimol string
- blob — A CACTVS ensemble block in zlib-compressed base64-encoded format
- key_cur — An identifier of an existing NCBI Netcache session. This will only work with sketcher installations which use NCBI Netcache to store state. NCBI Queueman sessions cannot be resumed for technical reasons.
- sid — A PubChem database record identified by its Structure identifier.
- cid — A PubChem database record identified by its Compound identifier.
- did — A PubChem deposition system record identified by its Deposition identifier. Access will only succeed if a cookie in environment variable HTTP_COOKIE contains the session ID of a registered depositor and when that session ID grants access to the specific deposition identifier. It is not possible to view arbitrary deposition IDs.
- qmid — A PubChem queue manager ID. This identifier is only used for the purpose of revisiting a previous query on the original PubChem system.
- vid — A PubChem structure upload vetting ID. This is a space-separated list of a Queueman request ID and the name of a blob associated with that request. Please remember that in parameter URLs the space character separating these two parts must be escaped as a plus sign. The content of the blob is a binary, un-compressed CACTVS structure blob. In contrast to all other initial sketcher content identifiers, this parameter establishes a continuous database connection. Every change of the loaded structure is automatically and immediately reflected in the Queueman database. All other initialization data identifiers are read-only, i.e. the PubChem database is not updated when a pre-loaded CID is edited in the sketcher.
- vhadd — This parameter is only useful in combination with a VID identifier. It is a boolean value which determines whether the structure written back into the database blob identified by

1. From Molinspiration, www.molinspiration.com

the VID parameter will undergo an automatic hydrogen addition step before it is stored or whether it is transmitted as it appears in the sketcher. The visible sketcher content is not changed regardless of the value of this parameter. If it is not set, the default value for this parameter is 0.

In case more than one of these parameters is specified, only the first one that is successfully decoded will be used. In case a preload identifier cannot be loaded, it is silently ignored.

It is common practice to construct the URL to open the sketcher window with contents of form fields and other dynamic data. Here is a sample JavaScript routine:

```
function startEditor()
{
  cid = document.forms["query"].elements["simple_cid"].value;
  smarts = document.forms["query"].elements["simple_searchdata"].value;
  if (cid!="") {
    editorwin = open("../edit/index.html?cid="+cid+"&cnt="+eventcnt, "editor",
      "height=440,width=880,scrollbars=no,status=yes,location=no,menubar=no,toolbar=no,resizable=yes",true);
  } else if (smarts!="") {
    smarts = encodeURIComponent(smarts);
    editorwin = open("../edit/index.html?smarts="+smarts+"&cnt="+eventcnt, "editor",
      "height=440,width=880,scrollbars=no,status=yes,location=no,menubar=no,toolbar=no,resizable=yes",true);
  } else {
    editorwin = open("../edit/index.html?cnt="+eventcnt,"editor",
      "height=440,width=880,scrollbars=no,status=yes,location=no,menubar=no,toolbar=no,resizable=yes",true);
  }
}
```

Note the encoding step for the SMARTS value which is needed to protect characters such as # which frequently occur in SMARTS strings but have special meaning within URLs.

Receiving Sketcher Data

The sketcher server transmits structure updates in a near-continuous fashion. There is no specific user act which initiates the transfer of sketcher content to a receiving form or other recipient.

At the installation time of a sketcher instance, the administrator selects a set of supported transfer formats. When a sketcher window connected to a server is running, it will attempt to transfer its current content in all installed formats to recipients. It does this by trying to call a sequence of JavaScript functions on the opener page with the current structure data in any of the installed encodings as argument. The names of these trans-

fer functions are fixed. A page using the sketcher as data provider simply needs to have one or more of these functions in its <*javascript*> page sections. Within these functions, page-specific custom JavaScript code can, for example, copy the received data to a form element and simultaneously perform any other related functions. The advantage of this generic mechanism is that the sketcher application does not need to know anything about the data destinations and storage mechanisms of any form it is sending data to.

It is neither required nor normal usage to write all possible transfer functions. If functions are missing, a failure to call them from the sketcher window is simply ignored.

This is a simple sample transfer function:

```
function transferSmiles(s) {
  document.forms["query"].elements["editor_smiles"].value = s;
  resetCIDref();
}
```

A caller page must use the same document domain as the sketcher installation it is referring to. This is explained in more detail in the next paragraph.

These are all possible transfer functions. A specific installation will usually only support a subset of these.

- transferSmiles(s) Receive the SMILES or SMARTS encoding of the current editor content. The encoding style is controlled by the option menu for setting the style of the data display above the drawing area in the sketcher window.
- transferBlob(s) Receive a CACTVS toolkit compressed base64-encoded serialized ensemble object. This is the only transfer mechanism guaranteed to be lossless, i.e. not to drop any query attributes or other structure attributes. On the other hand, these blobs are significantly larger than SMARTS strings and not conveniently displayed in an HTML form element. An alternative is to store these in a hidden form field, and to display the approximate SMARTS version in a visible form field. When the form is submitted, and the visible SMARTS text in the form has not been edited after it was received from the sketcher, the hidden blob can be transmitted as real query structure. One drawback is that this option about doubles the bandwidth requirements of the sketcher because of the frequent transfer of full blobs instead of short SMARTS strings. For this reason, some installations may not want to install this transfer option.
- transferMolfile(s) Receive a string image of an MDL Molfile, possibly with ISIS query data. Since these Molfile images are comparably large, this function is frequently disabled in Sketcher installations.

- transferJme(s) Receive a JME editor string of the current editor content
- transferInChI(s) Receive the InChI string of the current editor content
- transferSLN(s) Receive the current editor content as Sybyl Line Notation string
- transferMinimol(s) Receive a CACTVS V2 minimol in base64-encoding
- transferFormula(s) Receive the molecular formula, including implicit H
- transferKeys(k1,k2) Receive NCBI Netcache storage keys for the current and backup ensembles This is only meaningful if the state storage mechanism of the sketcher installation uses the NCBI Netcache daemon. In that case, sessions can be resumed using these keys. This is described in the paragraph on pre-loading the sketcher content.
- transferSessionID(id) Receive the session ID.

Document Domain Issues

The Internet domain any instance of the sketcher operates in is determined at installation time. In the simplest case, it is the same as that of the host it runs on, but it can be set to a more generic domain. For example, the sketcher may be installed on *pubchem.ncbi.nlm.nih.gov*, but it may be configured to run in domain *nlm.nih.gov*. A configured subdomain can only be a left-truncated part of the host domain, not an arbitrary domain, and has at least two levels left. For example, a sketcher installed at *pubchem.ncbi.nlm.nih.gov* can be configured to run in domains *pubchem.ncbi.nlm.nih.gov*, *ncbi.nlm.nih.gov*, *nlm.nih.gov* or *nih.gov*, but nowhere else.

It is important to know the domain a referenced sketcher is operating in, because the calling Web page must have its document domain set **exactly** to the same domain. This is because the sketcher needs to be able to call the data transfer functions in the opener window. If this window is in a different domain (even a more specialized domain of the sketcher domain), the function call will be blocked by the Web browser cross-site scripting security mechanisms, and no data from the sketcher ever arrives at the opener window.

The document domain of an HTML page is set by a simple JavaScript statement like

```
document.domain = 'nlm.nih.gov';
```

which should be executed directly at load time in the <*script*> section, or in an *onload()*-handler function attached to the <*body*> tag.

The domain modification only succeeds if the document domain of the page is already that domain, or a more specialized variant, such as *cis.ncbi.nlm.nih.gov*. It also means that no Web service outside the *nlm.nih.gov* domain (assuming the sketcher instance is installed there) can make use of the data transfer functionality of the sketcher, because it will not be able to set its document domain to the required value.

However, even from within a qualified domain, an opener page does not necessarily immediately possess a document domain which allows the domain adjustment to succeed. Depending on Web server configuration, the implicit domain may not have been resolved into a fully qualified path name. For example, if a HTML page which wants to link to a sketcher instance is called as *http::/cis/myapp/form.html*, the page domain as seen from JavaScript on that page may simply be *cis*, even if the cis computer is in the *nlm.nih.gov* domain and its fully qualified name is *cis.ncbi.nlm.nih.gov*. In that case the domain setting statement will still fail, and the editor no be able to transmit data to that page.

In order to solve this problem, a sketcher-dependent Web page should always check its own document domain and execute a reload with a fully qualified domain name if necessary. Here is a simple sample function to achieve this:

```
function checkDomain() {
    if (document.domain!='cis.ncbi.nlm.nih.gov') {
        if (!location.pathname) {
            location.replace("http://"+"cis.ncbi.nlm.nih.gov/"+location.search);
        } else {
            location.replace("http://"+"cis.ncbi.nlm.nih.gov/"+location.pathname+location.search);
        }
    }
    document.domain = 'nlm.nih.gov';
}
```

This function should only be called in as an *onload()*-handler attached to the <*body*> tag. Direct execution in the script section of the page is not safe. Many older Web browsers tend to crash or lock up if a *location.replace()* is executed while the original page is still loading.

The caller document domain should be kept constant during the time the editor window is open. The callback functions are called every time the editor content changes. If the document domain has been further modified since the window was opened, data transfer can fail mysteriously in the midst of an editing session because the license to call the transfer functions in the opener page has been lost.