

OpenEye
SCIENTIFIC

VIDA

Release 2026.1.0

Cadence Design Systems, Inc.

August 06, 2026

CONTENTS

1 Introduction **1**
- 1.1 General Concepts . . . 1
- 1.2 Scope . . . 2
- 1.3 Layout . . . 3
- 1.4 Undo / Redo . . . 6
- 1.5 Mouse Map . . . 6
- 1.6 User Directory . . . 6
- 1.7 Help . . . 7

2 Installation and Platform Notes **9**
- 2.1 Enterprise Installation for Windows . . . 9

3 File I/O **11**
- 3.1 Opening Files . . . 11
- 3.2 Closing Files . . . 14
- 3.3 Saving Files . . . 14
- 3.4 Importing Files . . . 18
- 3.5 Exporting Files . . . 18
- 3.6 Drag and Drop . . . 19
- 3.7 Copy and Paste . . . 19

4 3D Display **21**
- 4.1 User Interaction . . . 21
- 4.2 Rendering . . . 30
- 4.3 Stereo . . . 30
- 4.4 Viewpoint . . . 30
- 4.5 Bookmarks . . . 31
- 4.6 Tiled Display . . . 31
- 4.7 Toolbars . . . 31
- 4.8 Display Widgets . . . 32
- 4.9 Molecular Visualization . . . 36
- 4.10 Grid Visualization . . . 43
- 4.11 Grid Types . . . 43
- 4.12 Reentrant Grids . . . 45
- 4.13 Surface Visualization . . . 45
- 4.14 Symmetry . . . 48

5 2D Display **51**
- 5.1 Preview vs. Viewer . . . 52
- 5.2 Interaction . . . 52

5.3 Advanced Depiction Options . . . 53

6 Spreadsheet **55**
6.1 Basic Usage . . . 56
6.2 Sorting . . . 57
6.3 Displaying Data . . . 57
6.4 Organizing Columns . . . 58
6.5 Creating New Columns . . . 58
6.6 Filtering . . . 62
6.7 Statistics . . . 64
6.8 Formatting Columns . . . 65
6.9 Advanced Depiction Options . . . 67
6.10 Importing/Exporting . . . 69
6.11 Atom Spreadsheet . . . 72
6.12 Residue Spreadsheet . . . 72

7 List Management **73**
7.1 Introduction . . . 73
7.2 Browsing . . . 74
7.3 Views . . . 75
7.4 List Columns . . . 76
7.5 List Manipulation . . . 79

8 Processes **81**
8.1 Processes . . . 82
8.2 Pipes . . . 85

9 Molecule Manipulation **87**
9.1 Creating Molecules . . . 87
9.2 Editing Molecules . . . 91

10 Custom Views **99**
10.1 FRED View . . . 99
10.2 EON View . . . 101
10.3 ROCS View . . . 103

11 Selection Language **105**
11.1 Introduction . . . 105
11.2 Operators . . . 105
11.3 Lists and Ranges . . . 105
11.4 Properties . . . 106
11.5 Macros/Pre-defined sets . . . 107
11.6 Scripting with ScratchScope . . . 108

12 Scripting **111**
12.1 Journal File . . . 111
12.2 Startup File . . . 111
12.3 Scripts Directory . . . 111
12.4 Scripting API . . . 111
12.5 Scripting Window . . . 112
12.6 OpenEye Toolkits . . . 112
12.7 Customizing the User Interface . . . 113

13 Extensions **115**
13.1 Example Extensions . . . 115

13.2 Extension Manager . . . 116
13.3 Creating Extensions . . . 119
13.4 Distributing Extensions . . . 121

14 Preferences **123**

15 Menus **125**
15.1 File . . . 125
15.2 Edit . . . 130
15.3 View . . . 133
15.4 Data . . . 133
15.5 Style . . . 134
15.6 Bookmarks . . . 143
15.7 Tools . . . 144
15.8 Select . . . 144
15.9 Window . . . 146
15.10 Help . . . 148

16 Examples **149**
16.1 Data . . . 149
16.2 Scripts . . . 150

17 VIDA Python API **177**
17.1 General Functions . . . 177
17.2 Scope Functions . . . 191
17.3 User Interface Functions . . . 202
17.4 Display Functions . . . 241
17.5 Object Functions . . . 300
17.6 Molecule Builder Functions . . . 325
17.7 Data Analysis Functions . . . 340
17.8 Deprecated Functions . . . 355

18 Release History **367**
18.1 VIDA 5.0.7 . . . 367
18.2 VIDA 5.0.6 . . . 367
18.3 VIDA 5.0.5 . . . 367
18.4 VIDA 5.0.4 . . . 367
18.5 VIDA 5.0.3 . . . 367
18.6 VIDA 5.0.2 . . . 367
18.7 VIDA 5.0.1 . . . 368
18.8 VIDA 5.0.0 . . . 368
18.9 VIDA 4.4.0 . . . 368
18.10 VIDA 4.3.0 . . . 370
18.11 VIDA 4.2.0 . . . 372
18.12 VIDA 4.1.1 . . . 374
18.13 VIDA 4.1.0 . . . 374
18.14 VIDA 4.0.3 . . . 376
18.15 VIDA 4.0.0 . . . 377
18.16 VIDA 3.0.0 . . . 378
18.17 VIDA 2.1.2 . . . 379
18.18 VIDA 2.1.1 . . . 381
18.19 VIDA 2.1.0 . . . 382
18.20 VIDA 2.0.2 . . . 383

19 Citation **385**

20 Copyright and Trademarks **387**

21 Sample Code **389**

22 Citation **391**
22.1 Orion® . . . 391
22.2 Orion Floes . . . 391
22.3 Toolkits and Applications . . . 391
22.4 Publications for Bibliographies . . . 393
22.5 OpenEye MMDS Web Service . . . 394

23 Technology Licensing **395**
23.1 GCC . . . 467

Index **483**

CHAPTER

ONE

INTRODUCTION

1.1 General Concepts

There are some key concepts that are helpful for understanding how to interact with VIDA. Individual objects in VIDA can be in various states, including *Focused*, *Visible*, *Locked*, *Marked*, and *Selected*. Each of these states carries some special behaviors which are described below. These states also provide a way to refer to specific molecules or sets of molecules. For example, one could apply a coloring scheme or rendering style to all *Marked* objects, or to the *Focused* object.

VIDA's "Scope" determines which objects will be affected by operations within the program, based on their states, and is discussed in the "Scope" section below.

1.1.1 Focused State

The *Focused* state is a property of a single object in the application; there can be only one *Focused* object at any given time. Being *Focused* indicates that that particular object is the current object of interest and is therefore the primary focus of all relevant windows. If the *Focused* object is not already *Visible* (see below), it will behave as if it were *Visible* for as long as it remains *Focused*.

In addition, when a molecule is the *Focused* object, its SMILES representation is displayed in the application's status bar (which can be found at the bottom of the application window). If any of the molecule's atoms are selected, that selection will be indicated in the SMILES display by the coloring of relevant atoms. Selected atoms are colored using the current selection color (the default is orange) and are also in a bolded font while all of the unselected atoms will be displayed in their usual normal black font.

For large molecules (>100 atoms in this case), no SMILES representation is displayed as it would not fit in the status bar.

1.1.2 Visible State

The *Visible* state is an indicator of whether or not the associated object is drawn in the 3D display. Multiple objects can be *Visible* simultaneously. If VIDA is operating in tiled mode (see *Tiled Display*), each *Visible* object will be displayed in its own individual pane.

If the 2D display is also shown and VIDA is in tiled mode, *Visible* molecules will be drawn in the 2D display as well as the 3D display.

1.1.3 Locked State

The *Locked* state is essentially the same as the *Visible* state except in the situation where the application is in tiled mode, in which case each *Locked* object will be displayed in every pane of the 3D display. However, a *Locked* object can also be simultaneously *Visible* and/or *Focused* in which case it will be displayed in its own individual pane as well as every other pane in the 3D display.

1.1.4 Marked State

The *Marked* state is simply an indicator of user interest in the associated object. The *Marked* state can be used to help filter data as well as to specify the desired input to application functions. For instance, the *Marked* state can be used to specify which molecules out of a large list are saved. Finally, when certain operations generate too many results to be displayed at one time, the *Marked* state can be used to indicate the results of those operations.

1.1.5 Selected State

The *Selected* state is a special property of the *Visible* state. Only objects that are *Visible*, *Focused*, or *Locked* can be *Selected*. Much like the *Marked* state, the *Selected* state is an indicator of interest and provides an input set to application functions. Unlike, the *Marked* state, the *Selected* state is more transient and is easily cleared. The act of *selection* is usually performed in either the 3D or 2D display. Selected objects are usually indicated by a change of color (the default is orange) in the 2D display, and by a colored outline in the 3D display. More details on the actual process of selection can be found in the chapters on 3D and 2D displays.

1.2 Scope

Scope is a state of VIDA, and indicates which objects the application should operate on. The default application scope is *Focused* which means that all of the application functions will operate on the *Focused* object.

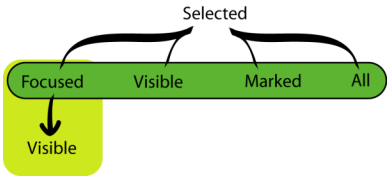


Fig. 1: **Scope Order of Precedence**

However, as seen in *Figure: Scope Order of Precedence*, if there is anything currently *Selected*, the selected set will take precedence over the current scope *i.e.* the action will apply only to the selected atoms/objects. Furthermore, in *Focused* scope, if there is no *Selected* set and no *Focused* object, the *Focused* scope defaults to behaving as if it were the *Visible* scope.

The *Visible* scope operates on all objects that are currently *Visible*, *Locked*, and/or *Focused*.

The *Marked* scope operates on all objects that are currently *Marked*.

The *All* scope operates on every object currently loaded in the application. Operating on the *All* scope can be a very lengthy process, and as such, its progress is displayed in the progress bar in lower right hand corner of the application. Operations performed on the *All* scope can be halted if desired by clicking on the *Stop* button immediately to the left of the progress bar.

The current scope is displayed in and can be modified from the main application toolbar as seen in the figure below.



Fig. 2: **Scope Toolbar Control**

1.3 Layout

The layout of the application should be relatively familiar to most users. VIDA provides a menu bar with many standard as well as specific menus, a toolbar for common operations, a central main window, and a number of peripheral windows. An example of the layout can be seen in *Figure: Docked Layout*.

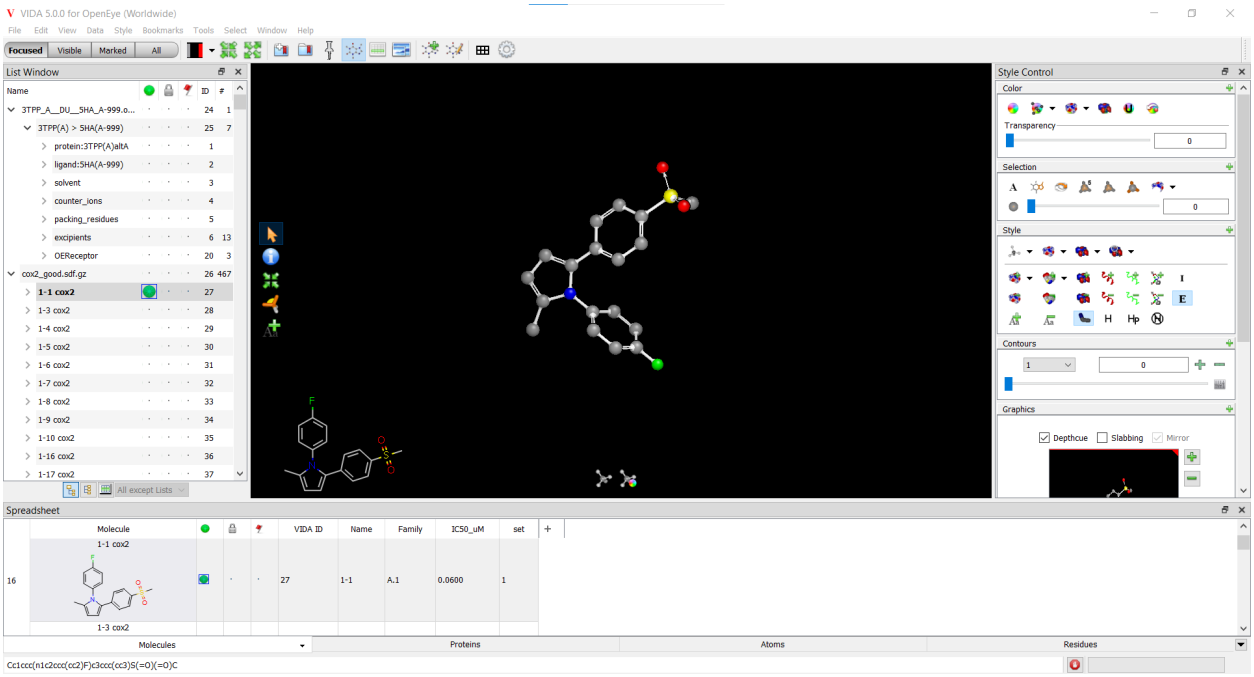


Fig. 3: **Docked Layout**

The layout including the position and visibility of the main and peripheral windows is saved on exit and will be restored the next time the application is run. The layout of the windows can be changed by clicking on the title bar of the individual window and dragging it to the desired location. A placeholder window will appear when the mouse is over an area where the dragged window can be placed as seen in *Figure: Dragging Windows*.

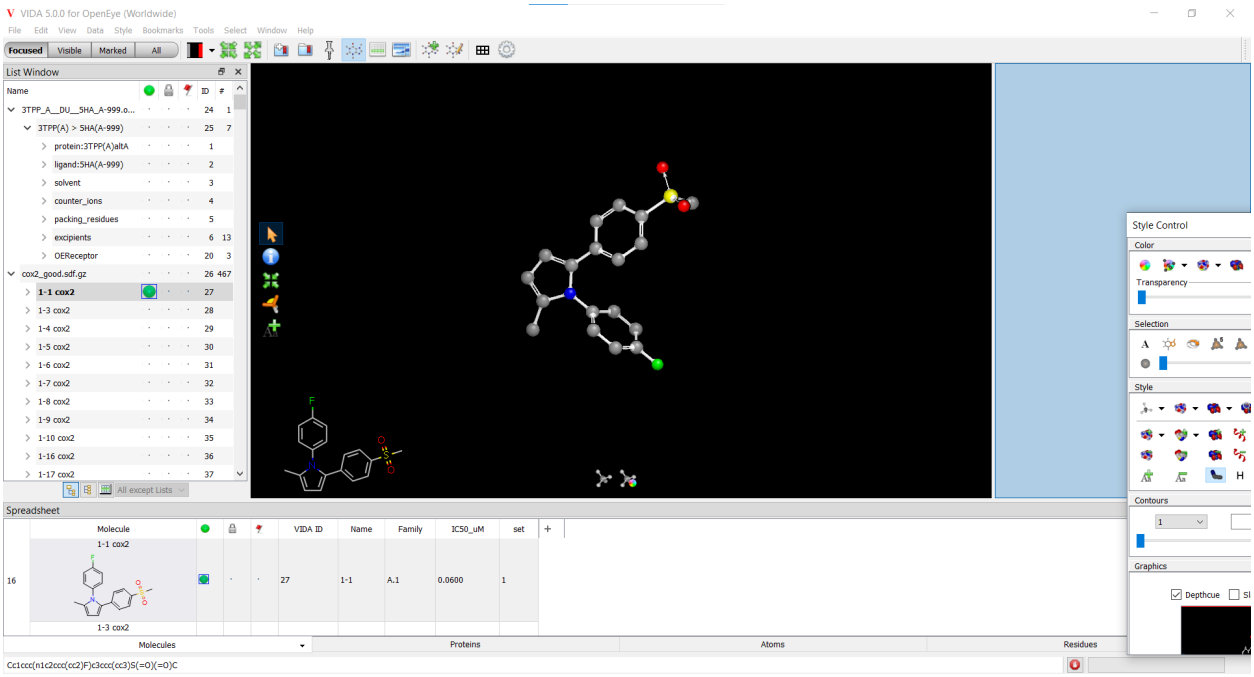


Fig. 4: **Dragging Windows**

Furthermore, multiple windows can be tiled on top of each other into a tab controlled area as can be seen in the bottom window in *Figure: Docked Layout*. This example shows tabs for "Spreadsheet" and "Scripting Window".

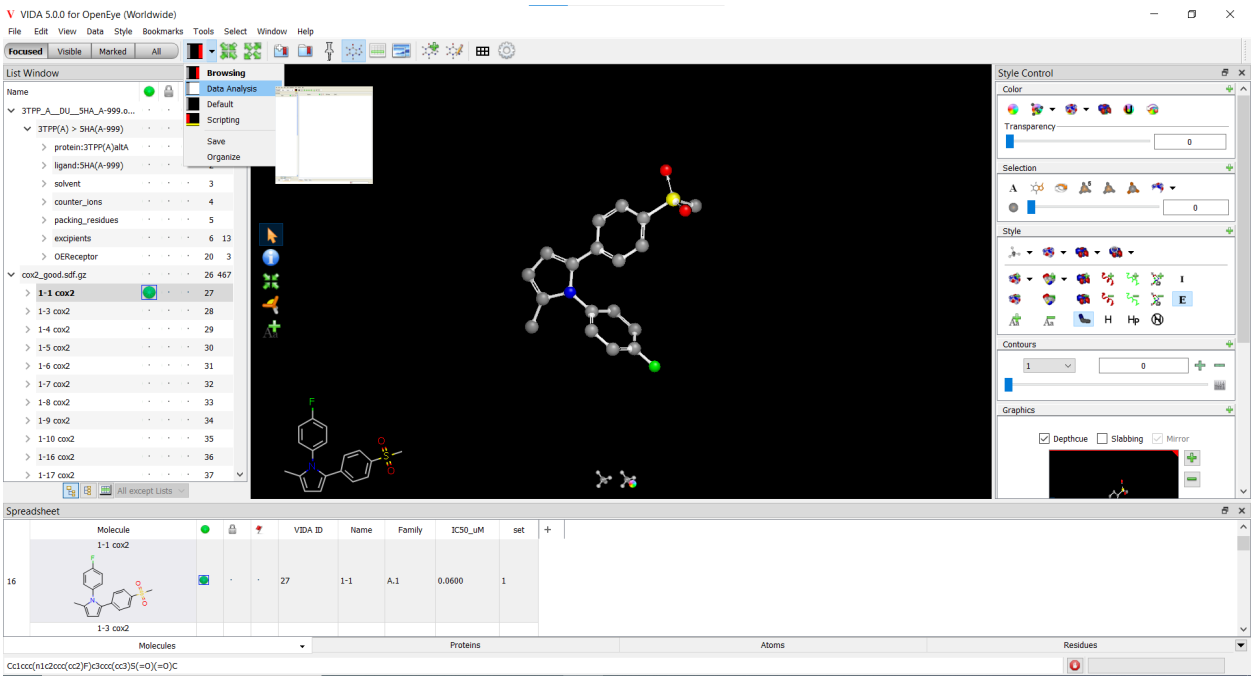


Fig. 5: **Custom Layouts**

In addition to manually changing the layout, there are a number of predefined layouts which are accessible via a drop down button on the toolbar as can be seen in *Figure: Custom Layouts*. The colored icon associated with each option

indicates which peripheral windows will be shown and where. Hovering the mouse over any given option for about one second will generate a preview image of the resulting layout. The "Save" option allows for the creation and storage of custom layouts. The "Organize" option enables the reorganization of the ordering of the layouts.

1.3.1 Main Window

The main window is the window that occupies the central area of the application on the screen. At any given time, there can be only one main window; however, there are multiple windows which are capable of being the main window if desired. The choice of main window can be controlled by selecting the desired option from the "Main window" submenu in the top-level *Windows* menu. In *Figure: Docked Layout*, the 3D display (see *3D Display*) is serving as the main window.

Currently, there are three different main window options: the 3D display, the 2D display, and the spreadsheet. More details about the individual windows can be found in their relevant chapters.

Fullscreen

In addition to the standard layout described above, the main window can optionally be displayed in fullscreen mode, by selecting the "Fullscreen" option in the top-level *View* menu. Pressing the "Esc" key will exit fullscreen mode (please note that clicking the mouse on the screen may be required before hitting the "Esc" key in order to ensure that the key press is registered).

When in fullscreen mode, the main menu bar and status bar are hidden by default. However, they can be toggled on or off by hitting the "F1" and "F2" keys respectively.

The 3D display normally contains popup toolbars around the edge of the window. These toolbars remain when in fullscreen mode to allow for easy access to display functionality. For more details on these toolbars see *3D Display*.

1.3.2 Peripheral Windows

In addition to the main window, there can be multiple peripheral windows docked around the edges of the main window or as separate floating top level windows. Furthermore, multiple windows can be placed on top of each other in a tabbed region.

Currently, multiple peripheral windows (including the potential main windows) are available:

- 2D Display
- 2D Preview
- 3D Viewer
- Builder
- List Window
- Scripting Window
- Spreadsheet
- Style Control

The display of peripheral windows is controlled by toggling the desired window options in the *Windows* menu. Currently visible windows are indicated by the presence of a check mark next to their names. Peripheral can also be hidden by clicking on the 'X' in the upper right hand corner of the window. As mentioned above, the layout of the peripheral windows can be controlled by dragging the individual windows to their desired locations (see *Figure: Dragging Windows*).

1.4 Undo / Redo

VIDA provides multi-step undo and redo functionality for almost all operations. Operations can be undone by selecting the "Undo" option in the *Edit* menu, clicking on the undo button in the top 3D toolbar, or by hitting *Ctrl+Z* on the keyboard. Any of these actions will undo the last operation performed. This can be done repeatedly or the undo history can be viewed in the "Undo History" option the *Edit* menu which allows selecting how far back into the undo history to go. Please note that while this menu only displays the 10 most recent operations, the undo history can be much greater, so feel free to revisit the "Undo History" option to go back further.

Operations that were just undone can be redone by selecting the "Redo" option in *Edit* menu, clicking on the redo button in the top 3D toolbar, or by hitting *Ctrl+Y* on the keyboard. Any of these will redo the last operation undone. This can be done repeatedly or the redo history can be viewed in the "Redo History" option in the *Edit* menu which allows selecting how far into the redo history to go. Like the undo history, this menu only displays the 10 most recent undone operations, but the redo history may be much greater and can be revisited to go further.

1.5 Mouse Map

User interaction in the 3D display is primarily performed using the mouse. The following table details the available functions as well as the mouse actions required to perform those functions.

Function	Mouse Action
Zoom	Scroll wheel
Zoom	Move mouse with *Middle* button
Zoom	Move mouse with *Left* + *Right* buttons
Rotate	Move mouse with *Left* button
Rotate around Z axis	Move mouse with *Left* button + *Alt* key
Translate in XY plane	Move mouse with *Left* button + *Shift* key
Translate in Z plane	Scroll wheel with *Alt* key
Select	Click *Left* button
Select rectangle	Move mouse with *Right* button
Select add	Click *Left* button with *Ctrl* or *Shift* key
Select grow	Double click *Left* button
Select add grow	Double click *Left* button *Ctrl* or *Shift* key
Select rectangle and add	Move mouse with *Right* button and *Ctrl* or *Shift* key
Change clipping plane	Scroll wheel with *Ctrl* + *Shift* key
Change far clipping plane	Scroll wheel with *Alt* + *Ctrl* key
Change near clipping plane	Scroll wheel with *Alt* + *Shift* key
Text scale	Scroll wheel with *Ctrl* key
Menu	Click *Right* button
Adjust contour level	Scroll wheel with *Shift* key
Show labels	Move with *Ctrl* key

1.5.1 Special Builder Mouse Actions

When a molecule is in *Editing* mode, the **Rotate** and **Translate in XY plane** actions above can be applied exclusively to the molecule being edited by holding down the *Ctrl* key (*Command* key on a Macintosh) while rotating or translating.

1.6 User Directory

VIDA stores user specific information in a user writable local directory on the computer running the application. Contained in this directory are the user's application settings (*VIDA.ini*), preferences (*preferences.oeb*), journal file (*journal.py*), and a cached copy of the license file (*oe_license.txt*).

The location of the user directory is specific to the operating system, the details of which are listed below:

Linux / Unix

`~USERNAME/.OpenEye/VIDA/5.0`

macOS

`/Users/USERNAME/Library/OpenEye/VIDA/5.0`

Microsoft Windows 10

`C:\Users\USERNAME\AppData\Local\OpenEye\VIDA\5.0`

The user directory can be accessed from within VIDA by clicking on the "Open User Directory" option within the *Help* menu.

1.7 Help

Help with VIDA can be found under the *Help* menu in the main menubar, which will link to the documentation on the OpenEye website: https://docs.eyesopen.com. The *Help* menu provides an *About* option, access to documentation, licensing utilities, access to *Examples*, access to the *User Directory*, as well as the ability to file a bug report.

1.7.1 About

The *About* box provides extensive details about the running version of VIDA as well as the system on which that VIDA is running. It also provides licensing information and direct access to the license file in use.

CHAPTER

TWO

INSTALLATION AND PLATFORM NOTES

Follow the standard instructions for installation, licensing, and removal.

If a valid license is not found in one of the default locations, the user will be prompted to specify the location of the license file. The user may specify a local file, a file on a network/shared drive, or a network URL. If the specified file contains a valid license, the license file will be copied to a local cache so that it does not have to be found each time.

When the license file expires, a new license will automatically be searched for in the location that was specified for the current license. If a valid license is found in that location, it will automatically by copied over and used without any user action required. If a valid license file cannot be found, the user will again be prompted to locate one. When there are fewer than 30 days remaining on the current license, the application title bar will change to report the number of days remaining.

Once VIDA has started, the user can view the actual license file being used (*that is*, the locally cached copy), specify a new license file, or force an update of the current license from the originally specified location. These options can be accessed in the `License` submenu of the main `Help` menu.

2.1 Enterprise Installation for Windows

It is possible to perform a "silent installation" of VIDA by running the installer with the `/S` flag. During a silent installation, no progress or other installation windows will be displayed. If this flag is used, the installer should be run from an administrative account to avoid the appearance of a "User Account Control" dialog during installation. No reboot is required after installation.

By default, the OpenEye file extensions ".oes" (VIDA state files), ".oeb" (OpenEye binary files), and ".vpy" (VIDA Python scripts) are associated with the VIDA application. Users can optionally associate Mol files (.mol, .sdf, .sd), PDB files (.pdb, .ent), Mol2 files (.mol2), and grid and surface files (.grd, .agd, .srf, .oesrf) with VIDA, but by default these optional associations are not made.

CHAPTER

THREE

FILE I/O

3.1 Opening Files

There are multiple mechanisms by which a file can be loaded in VIDA. The simplest method is to select the *Open* option in the *File* menu. This will launch a file selection dialog which prompts the user for the desired file(s) which will subsequently be loaded. A short list of the most recently opened files can also be found in the *File* menu under the *Recents* (or *Open Recents* on Mac OS X) submenu. Selecting an option in this submenu will load the associated file. There is an *Open All* option at the bottom of the submenu. Files can also be loaded by specifying them on the command line when starting VIDA, by dragging the file of interest onto the VIDA desktop icon, or by dragging the file of interest into the *List Window*.

In addition to the *Open* menu item, there is a separate *Open State* item which filters out all files except for state files. State files can be opened using any of the mechanisms described above, but their special nature warrants its own option.

There is also an *Open Special* submenu which contains three default options: *Script*, *From PDB*, and *MTZ*. The *Script* option prompts for a file, but filters out all files except for Python scripts. The *From PDB* option prompts for a PDB ID and then fetches the associated molecule directly from the PDB. The *MTZ* option prompts for a crystallographic MTZ file, and then allows the user to select the maps and fields to be loaded from the file. The *Open Special* submenu also serves as an excellent target area for adding custom open operations via the scripting interface.

3.1.1 Molecules

Multiple molecular file formats are supported for reading and include :

- OpenEye binary format (version2: .oeb)
- OpenEye design unit format (.oedu)
- MDL, RDF, and SDF
- Tripos Mol2, Mol2H
- Daylight SMILES, canonical SMILES, and isomeric SMILES
- ChemDraw CDX
- ISIS Sketch
- MacroModel
- MOPAC
- PDB, MMCIF
- XYZ

As is often the case with file formats, the meaning and use of certain fields within a format may change over time which can potentially lead to problems interpreting those files. VIDA, using OpenEye's **OEChem** toolkit, makes its best effort to interpret all files correctly; however, it does provide a mechanism to allow the user to override the handling of

certain formats (SMILES, PDB, Mol2, XYZ, and MacroModel). The ability to change the *flavor* of a specific format can be done in the application preferences. *Figure: Molecule flavors*.

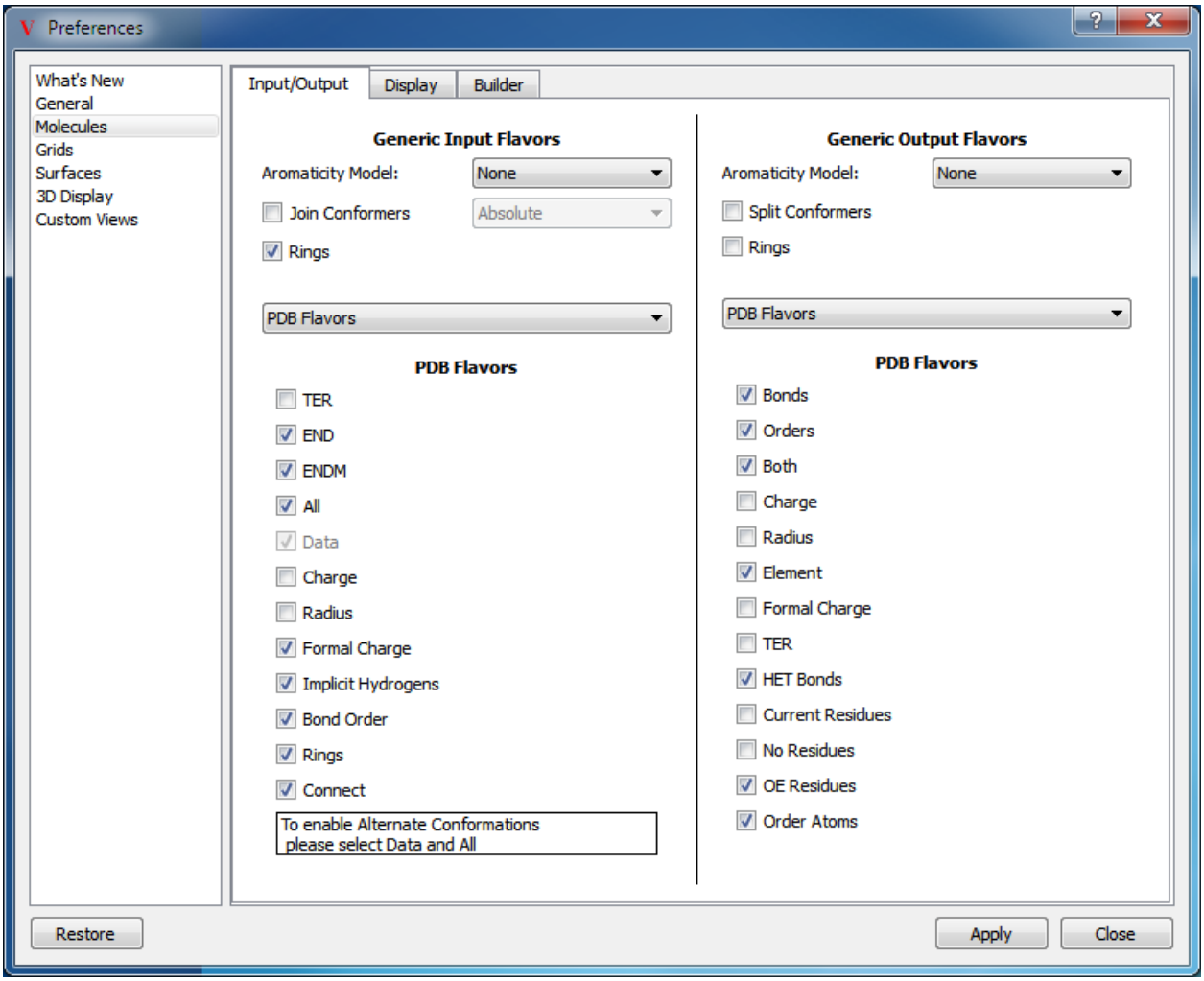


Fig. 1: *Molecule flavors*

An important flavor to be aware of is the *Data* flavor of the PDB format. Enabling this flavor allows VIDA to properly handle alternate residue locations specified in the PDB file.

In addition to changing the flavor of a format, there are a few other advanced options available when reading molecules including aromaticity model specification and conformer joining. The desired aromaticity model to be applied can be specified in the pull down menu next to the *Aromatic Model* label. The *Join Conformers* checkbox controls whether or not adjacent molecules in an input file will be tested on reading to determine whether they are unique compounds or simply different conformers of the same molecule. The specific test to be performed can be specified in the pull down menu next to the checkbox. For more specific details on the available aromaticity models and conformer tests, please see the **OEChem** documentation.

Note

Because these advanced options are available through the application preferences, they will be remembered by VIDA and automatically applied in future *Open* operations unless they are subsequently modified or restored.

3.1.2 Grids

Multiple grid and map formats are supported for reading and include:

- OpenEye grids
- ASCII grids
- GRASP/DelPhi grids
- CCP4 maps
- XPLOR maps
- MTZ maps

MTZ maps are a little special in the sense that the contents of the files do not follow a standard format and the relevant data columns for reading phases and amplitudes can be named arbitrarily. By default, VIDA attempts to find common column names that are used by standard refinement packages.

If a **MTZ** file cannot be opened, an error message is displayed suggesting the use of the menu item *File/Open Special/MTZ...*

To have more control over which maps are created, use the *File* menu option *Open Special/MTZ...*. This opens a dialog that allows selection of both the MTZ columns and the desired output maps.

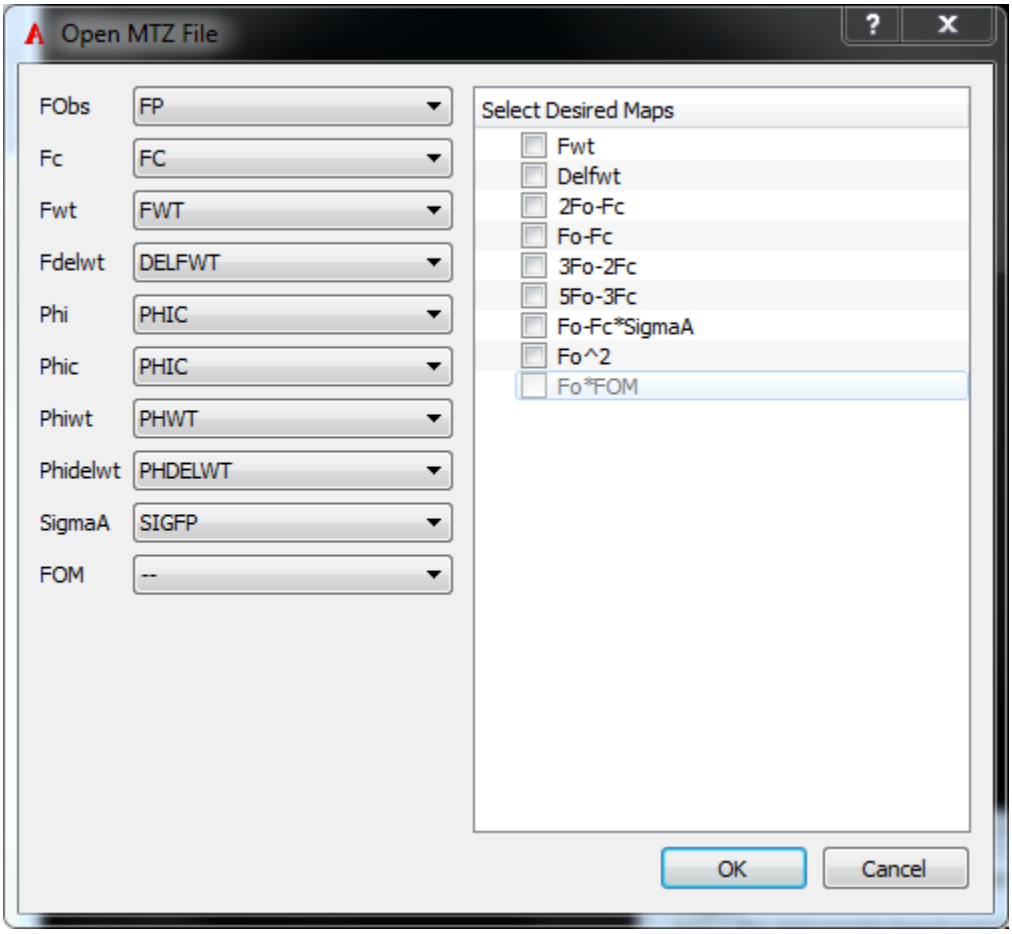


Fig. 2: *MTZ Open Dialog*

Simply select the appropriate columns and map types to generate and click “OK”.

3.1.3 Surfaces

Multiple surface formats are supported for reading and include:

- OpenEye surfaces
- GRASP surfaces

3.1.4 State Files

The entire state of a session can be stored in a single file called a *State File* (.oes). A state file contains all of the molecules, grids, and surfaces that were loaded in VIDA at the time the state file was created. In addition, the state file preserves the actual view, layout, and properties of that session. State files provide an extremely convenient way to save sessions for later work or to share with collaborators. Furthermore, state files are the fastest method of reading and writing large data sets.

To load a state file, choose the *Open State* option in the *File* menu and then select the desired state file. State files can also be loaded from the command line. It is important to note that loading a state file during a run will clear the current state before loading the new one. Therefore, be sure to save the current state before loading a new state if keeping the current state is desired. Furthermore, since state files contain the entire state of the application that generated them, loading a state file will overwrite local preferences; however, they will not be saved on exit.

Note

State files are not backwards compatible with older versions of VIDA

3.1.5 Python Scripts

VIDA supports reading of Python scripts compatible with Python version 3.7+. VIDA supports two additional file extensions for Python scripts (.pyv and .vpy) to enable users to associate those file extensions with VIDA without adversely affecting the association of unrelated Python scripts.

For more details about Python scripting in VIDA, see the chapter on *Scripting* as well as the associated API manual.

3.2 Closing Files

When a file is opened in VIDA, its contents are loaded into its own individual list. To close a given file, simply right-click on the associated list in the *List Window* and select the *Delete* option. This will close the file and remove the contents from the application. However, if an object is present in multiple lists, deleting one of those lists will not delete that object from the other lists. It is important to note that deleting a list, does not affect the actual file that was read.

To close all of the currently loaded files, select the *Clear All* option in the *File* menu. In addition to clearing everything that is loaded, this option effectively restores the state of the application to what it was when the application was started.

3.3 Saving Files

VIDA is capable of saving multiple types of files including molecules, grids, surfaces, and state files. The details of the individual formats can be found in the relevant sections below.

Note

VIDA does not currently support writing to oedu file format

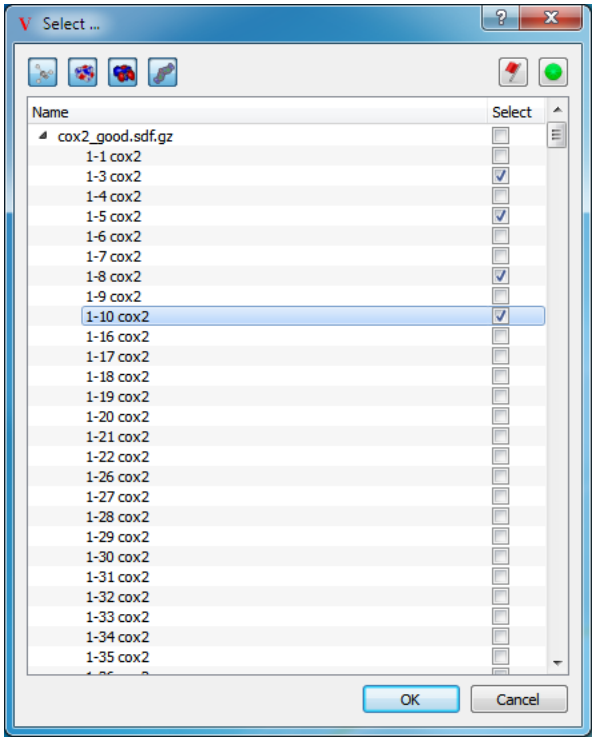


Fig. 3: **Prompt dialog**

A file can be saved by choosing the *Save* option in the *File* menu. This will launch a dialog prompting the user to select which objects to save (see *Figure: Prompt Dialog*). There are two sets of buttons at the top of this dialog which provide mechanisms for filtering the available options. The three buttons on the top left toggle the display of molecules, surfaces, and grids respectively. The two buttons on the top right provides filters for showing only the *Marked* and/or *Visible* objects respectively. Once one or more objects have been selected and the **OK** button is pressed, a file dialog will appear allowing the user to specify the desired file. The file formats listed in the file dialog filter will reflect which file formats are supported based on the objects selected. Only the OpenEye binary format supports writing multiple object types (molecules, grids, and/or surfaces) to the same file.

This process can be expedited by simply right-clicking on the desired objects and selecting the *Save* option. This allows the user to bypass the selection dialog and go directly to the file dialog.

3.3.1 Molecules

Multiple molecular file formats are supported for writing and include:

- OpenEye binary format v2 (.oeb)
- MDL, RDF, and SDF
- Tripos Mol2, Mol2H
- Daylight SMILES, canonical SMILES, and isomeric SMILES
- ChemDraw CDX

- MacroModel
- Molecular Formula (MF)
- MOPAC
- FASTA
- PDB
- XYZ

As is often the case with file formats, the meaning and use of certain fields within a format may change over time which can potentially lead to problems interpreting those files. VIDA, using OpenEye's **OEChem** toolkit, writes all files according to the standard; however, it does provide a mechanism to allow the user to override the handling of certain formats (SMILES, SDF/MDL, PDB, MOL2, MF, MOPAC, and MacroModel). The ability to change the "flavor" of a specific format can be done in the application preferences as seen in *Figure: Molecule flavors*.

In addition to changing the flavor of a format, there are a few other advanced options available when writing molecules including aromaticity model specification and conformer splitting. The desired aromaticity model to be applied can be specified in the pull down menu next to the *Aromaticity Model* label. For more specific details on the available aromaticity models, please see the **OEChem** documentation.

Note

Because these advanced options are available through the application preferences, they will be remembered by the application and automatically applied in future *Save* operations unless they are subsequently modified or restored.

3.3.2 Grids

Multiple grid formats are supported for writing and include:

- OpenEye grids
- ASCII grids
- GRASP/DelPhi grids

3.3.3 Surfaces

Multiple surface formats are supported for writing and include:

- OpenEye surfaces
- GRASP surfaces

3.3.4 State Files

The entire state of a session can be stored in a single file called a *State File* (.oes). A state file contains all of the molecules, design units, grids, and surfaces that were loaded in VIDA at the time the state file was created. In addition, the state file preserves the actual view, layout, and properties of that session. State files provide an extremely convenient way to save sessions for later work or to share with collaborators. Furthermore, state files are the fastest method of reading and writing large data sets.

Note

State files are not backwards compatible with older versions of VIDA

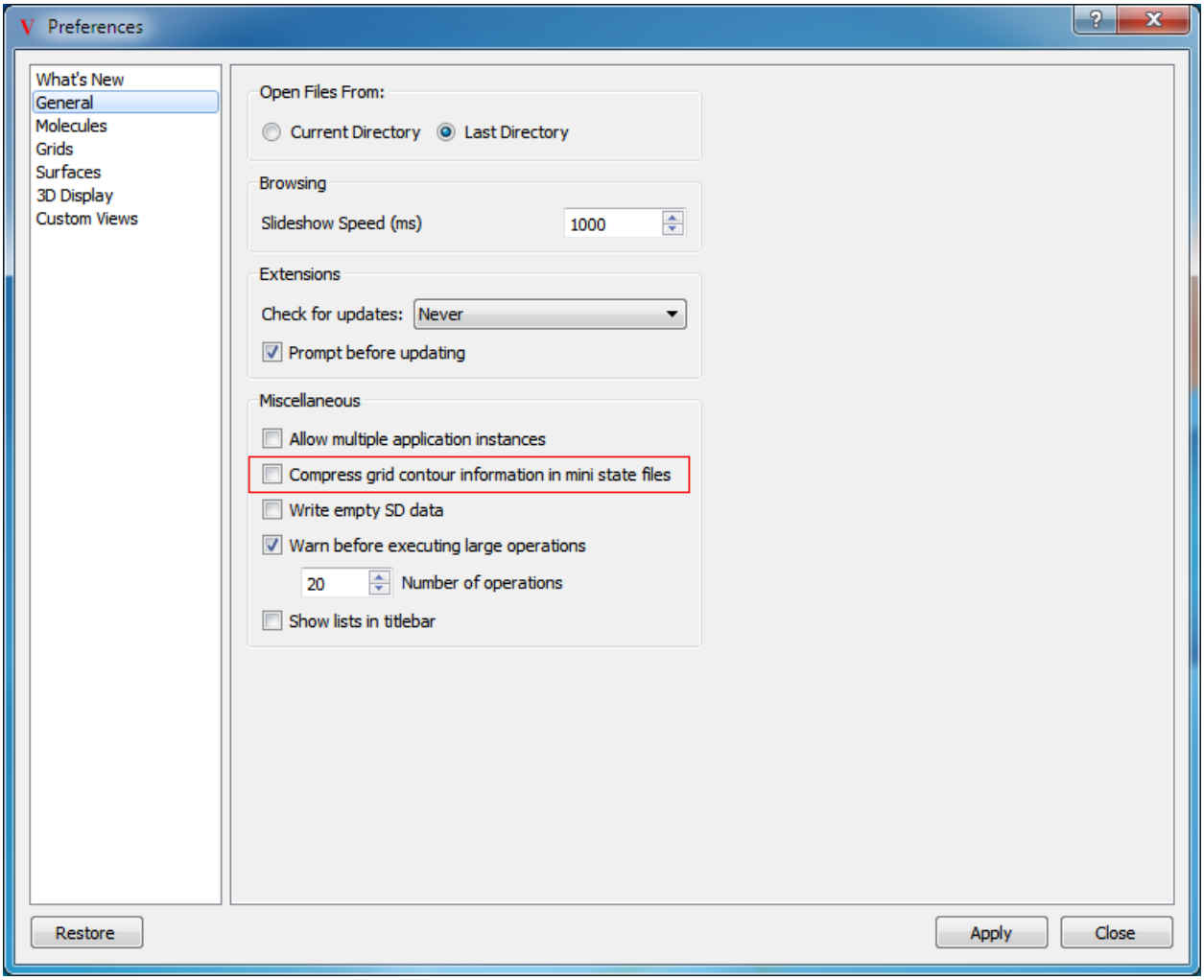


Fig. 4: **Compress Grid Contours**

VIDA supports a "mini" state file option which only writes out objects that are currently visible or referenced in the display bookmarks. This can provide for substantial savings in disk size and load time. This mini-state can be further compressed by selecting the "Compress grid contour information" option in the *General* section of the application preferences *Figure: Compress Grid Contours*. Selecting this option zeroes all non-contour related positions in the grid which greatly improves its compressibility. This should only be done if there is no expectation that the user loading the resulting state file will want to look at grid contour levels other than those already defined when the file was generated.

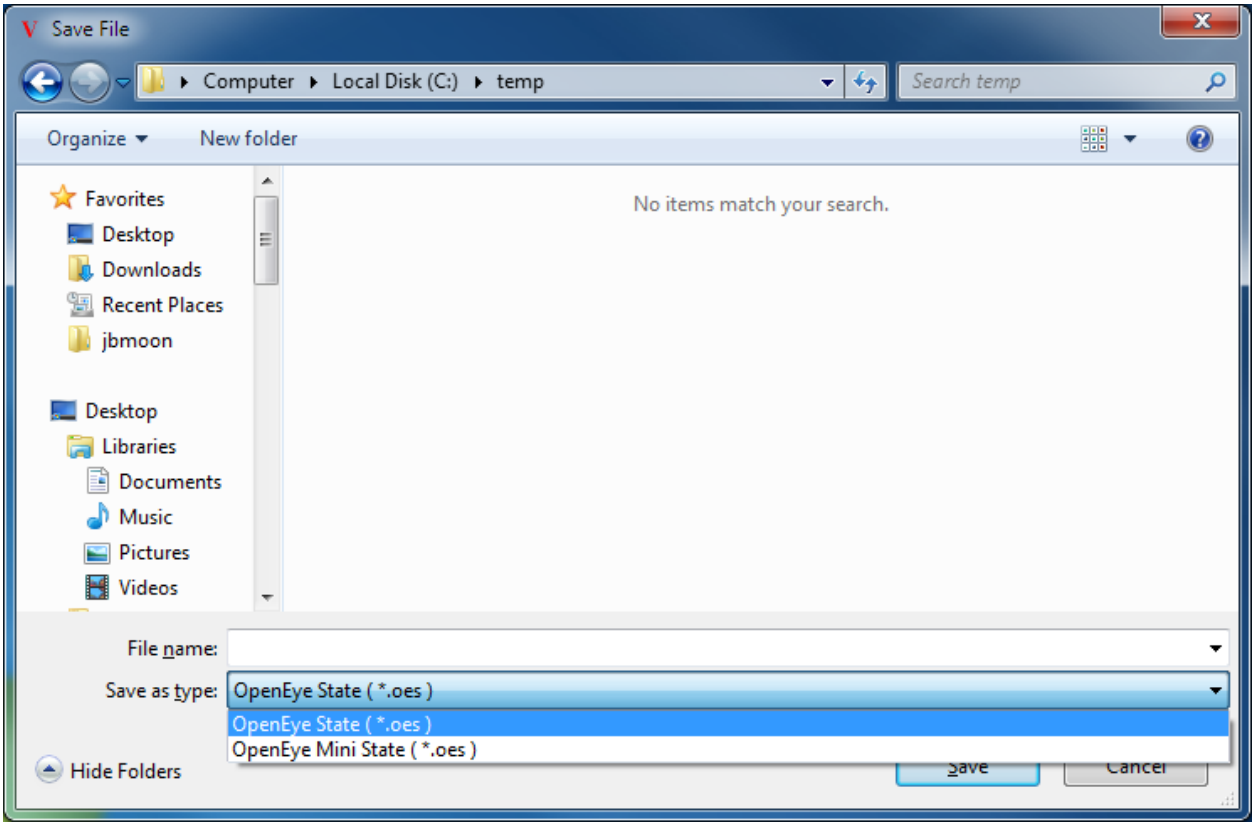


Fig. 5: **Saving State**

To save a state file, select the *Save State* option in the *File* menu and specify the desired state file. The state file type can be selected using the file format filter as seen in *Figure: Saving State*.

3.4 Importing Files

External spreadsheet data can be imported into VIDA and associated with currently loaded objects. Both comma separated files (.csv) and tab delimited files (.txt) are supported by default. Once a file has been selected, an import dialog will appear which allows the user to customize the column delimiters, column headers, as well as the criteria on which rows are matched to currently loaded objects. For more details on importing spreadsheet data, please see the Spreadsheet chapter.

3.5 Exporting Files

Multiple special file types can be exported from VIDA including data, images, and scripts. Exact details about these files can be found in the relevant sections below.

3.5.1 Data

Spreadsheet data can be exported in three formats: comma separated files (.csv) , tab delimited files (.txt), and PDF (.pdf). Data can be exported by selecting the "Spreadsheet" option in the *Export* submenu in the top-level *File* menu. Selecting this option will launch a file dialog which allows the user to specify the output file. Once the desired file has been chosen, another dialog will appear which allows the user to select which spreadsheet and which columns in that spreadsheet should be exported. For more specific details on exporting spreadsheet data please see the Spreadsheet chapter.

3.5.2 Images

POV-Ray

The current scene in the 3D display can be exported to a POV-Ray input file which can be used to generate very high quality and resolution images. To export a POV-Ray file, select the *POV-Ray* option in the *Export* submenu in the *File* menu. The application will then prompt the user to specify a file to which the scene will be saved.

3.5.3 Python

A complete history of the application's operations can be written out as a Python script. This script can be exported by selecting the *Script History* option in the *Export* submenu in the *File* menu.

3.6 Drag and Drop

A powerful drag and drop interface is provided for passing molecules, images of molecules, and molecular data between applications. Multiple mechanisms of transferring data via the drag and drop interface are supported.

- Dragging a list of filenames into the List Window will cause those files to be loaded.
- Dragging a list of SMILES into the List Window will cause those SMILES to be parsed into new molecules and stored in a new list called "Pasted". These new molecules will not be assigned 3D coordinates and therefore will not be visible in the 3D display.
- Dragging a list of IUPAC or common chemical names into the List Window will cause those names to be parsed into new molecules (using OpenEye's **Lexichem** toolkit) and stored in a new listed called "Pasted". Please note, this functionality requires a separate license for the **Lexichem** toolkit available from OpenEye. These new molecules will not be assigned 3D coordinates and therefore will not be visible in the 3D display.
- Dragging molecules from other molecular visualization programs into the List Window will cause those molecules to be loaded into a new list called "Pasted". The ability to load molecules from other applications depends on how the other applications place molecules on the clipboard. Currently supported clipboard formats include SMILES, ChemDraw CDX, ISIS Sketch, MDL SD Files, and OpenEye OEB files. Furthermore, for other molecular visualization programs that support a drag and drop interface, molecules can be dragged from the List Window into those applications.
- Dragging molecules from the List Window into other applications will cause molecules, molecular data, or molecular images to be transferred to that application depending on how that application expects to receive data. For instance, dragging molecules into a text editor will enter a list of SMILES corresponding to the dragged molecules.
- Dragging a 2D depiction from the Spreadsheet will cause that image to be loaded in the associated application if possible.
- Dragging molecules within the List Window can be used to organize files and molecules. See the *List Management* section for more details.
- DesignUnits cannot be modified with Drag-and-Drop at this time

3.7 Copy and Paste

A copy and paste interface exists for passing molecules, images of molecules, and molecular data between applications. The *Copy* option can be found in the *Edit* menu and can also be performed by pressing *Ctrl+C* on the keyboard. This will copy all of the molecules in the current scope to the clipboard, unless the active window is the Spreadsheet in which case it will copy all of the selected Spreadsheet data to the clipboard instead.

The *Paste* option can be found in the *Edit* menu and can also be performed by pressing *Ctrl+V* on the keyboard. This will paste any molecules found on the clipboard into a new list called "Pasted".

CHAPTER

FOUR

3D DISPLAY

The 3D display is the primary visualization interface available to the user (although it is not the only one, there is also a spreadsheet view and 2D views). The purpose of the 3D display is to allow the user to view and interact with molecules, grids, and surfaces.

4.1 User Interaction

The primary interface mechanism to the 3D display is the mouse which is discussed in detail below. However, the large number of potential operations that can be performed in the 3D display necessitate an additional mechanism of interaction besides the simple use of mouse buttons and popup menus. For this reason a separate peripheral window is provided to access these operations. This window is called the *Style Control* and is discussed below.

4.1.1 Mouse

As described above, the mouse is considered to be the primary interface mechanism to the 3D display. The details of how to use the mouse to interact with the 3D display are listed here. A three (or two) button mouse is recommended for use; however, support for a single button mouse is available on macOS. Holding down the *Ctrl* key while clicking or pressing the mouse button will emulate using the *Right* mouse button on a three button mouse.

For users not comfortable or familiar with the mouse interactions described below, VIDA can be made to emulate the mouse behavior of many other applications including: Coot, Insight, Maestro, MOE, O, Quanta, PyMol, RasMol, and SYBYL. The desired mouse map can be set in the application preferences in the *3D Display* section (*Figure: Mouse Map Preferences*). Please note that not all the functionality described below is available in the emulated mouse modes, nor is all of the functionality of the emulated applications available in VIDA.

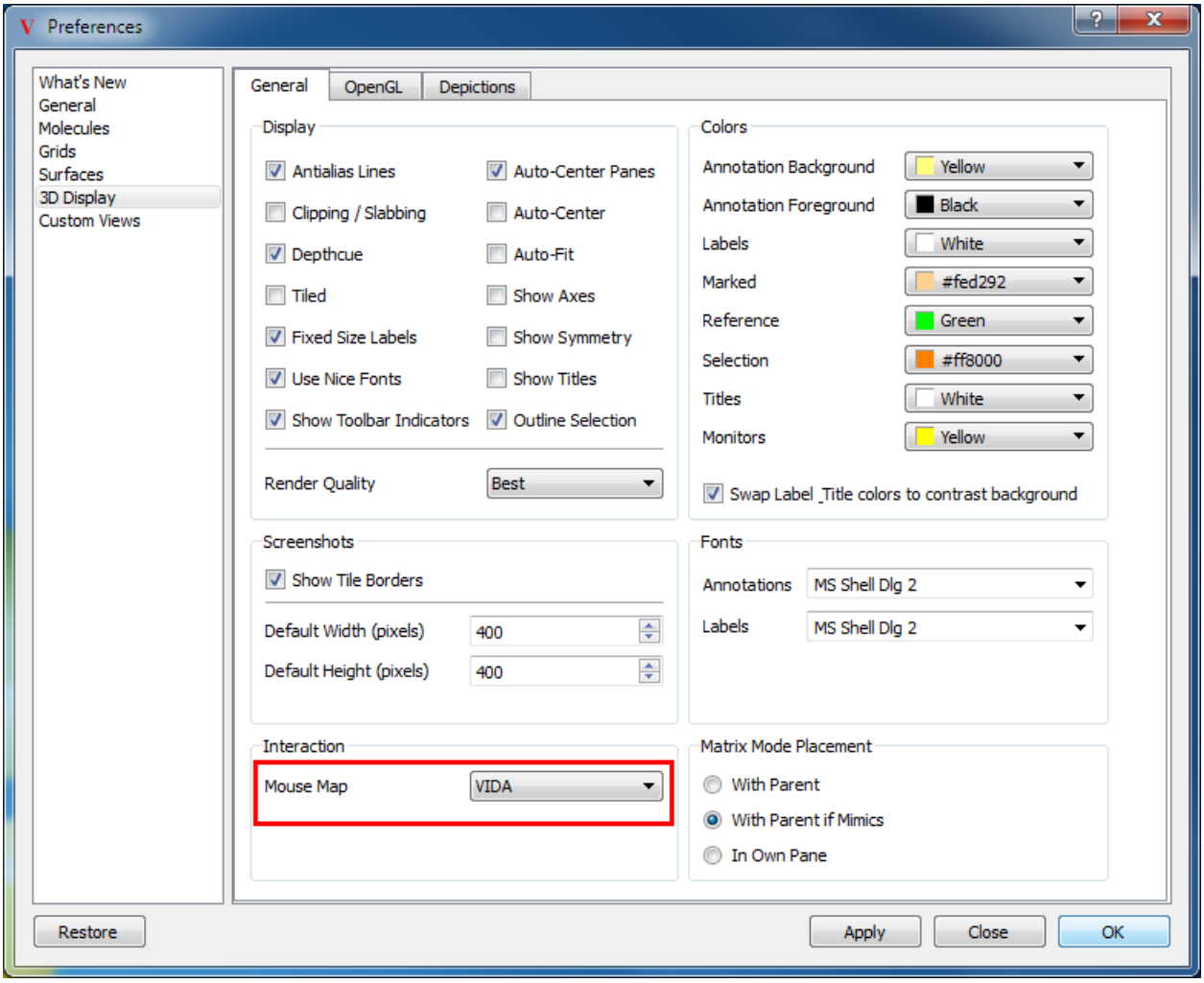


Fig. 1: **Mouse Map Preferences**

4.1.2 Selection

Objects in the display can be selected by clicking on them using the *Left* mouse button. The selection is cleared between button clicks (or by clicking in the background) unless the *Shift* or *Ctrl* key is also held down when clicking. If a single atom on a surface has been selected, holding down the *Shift* key and selecting a second atom will select the shortest path of vertices between the two selected end points.

Double-clicking using the *Left* mouse button expands the current selection to the next logical grouping. For instance, double-clicking on a selected atom in a protein will expand the selection to include all of the atoms in the same residue as the original selected atom. Double-clicking on the selected set again will expand the selection to include either the entire chain if there are multiple chains, otherwise it will include the entire molecule.

Selection can also be performed using a lasso style selection method by holding down the *Right* mouse button and moving the mouse in the window to define a selection rectangle (a dotted rectangle will be displayed on the screen as you do this). Releasing the *Right* mouse button will select all of the atoms and bonds inside of the rectangle. The previous selection will be cleared unless the *Shift* or *Ctrl* key is also held down while moving the mouse. Lasso style selection of surfaces and grids is not currently supported.

Additional methods of selection are available in the *Select Menu* pane of the *Style Control* and the *Select Menu*.

Rotation

The scene can be rotated in three dimensions by holding down the *Left* mouse button and moving the mouse in the window. These movements will emulate rotation using a trackball.

The scene can also be rotated around just the Z axis by holding down the *Alt* key and the *Left* mouse button simultaneously.

Translation

Using the mouse, the scene can be translated in the XY plane by holding down the *Shift* key and the *Left* mouse button simultaneously. Translation along the Z axis can be performed by holding down the *Alt* key, *Shift* key, and the *Left* mouse button simultaneously. The scene can also be translated along the Z axis by holding down the *Alt* key while using the mouse wheel.

Using the keyboard, the scene can be translated horizontally in the display by pressing either the *A* key or *D* key which translate to the left and to the right respectively.

Scale/Zoom

The scene can be scaled (or zoomed) by holding down the *Middle* mouse button by itself, the *Left* and *Right* mouse buttons together, or by using the mouse wheel. Any of these three modes will scale the scene. Multiple modes are provided for convenience and to accommodate the many varied mouse configurations in existence.

The scene can also be scaled using the *W* and *S* keys on the keyboard.

Text Scale

The scale of the text displayed in the scene can be controlled by using the mouse wheel while holding down the *Ctrl* key.

Clipping

The position of the near and far clipping (or slabbing) planes can be controlled using the mouse wheel while holding down both the *Ctrl* and *Shift* keys. Both planes are moved simultaneously and mirror each other's position. It is important to note that slabbing must already be enabled for these operations to actually be performed.

The clipping planes can be adjusted independently by holding down the *Alt* key with one of the *Ctrl* or *Shift* keys corresponding to the far and near clipping planes respectively.

Contour Level

The contour level of the grids in the default scope can be adjusted using the mouse wheel while simultaneously holding down the *Shift* key. Each incremental turn on the mouse wheel corresponds to an increase or decrease in the contour level by 0.1.

Label

Informative labels can be displayed about atoms and bonds (as well as surfaces and grid contours) underlying the current mouse position if the *Ctrl* key is pressed while moving the mouse (no mouse button need be depressed) on most platforms. On the Mac, this behavior is obtained by holding down the *Alt* key instead of the *Ctrl* key.

Additional methods of labeling atoms and bonds are available in the *Style* pane of the *Style Control* .

Menu

A context sensitive popup menu can be generated by clicking in the window while holding the *Right* mouse button down. This menu will contain operations that can be performed on those objects associated with the context when the popup menu was generated.

4.1.3 Style Control

The Style Control is a separate peripheral window which is provided to enable additional interactions with the 3D display. The Style Control is a vertical window consisting of many individual panes. Each pane provides its own set of functionality. The display of the individual panes can be toggled by clicking on the title bar of the pane of interest. In addition, individual panes can be "torn off" to act as top-level windows by clicking on the "+" icon in the top right of the title bar. When the pane is torn off the "+" icon becomes a "-" icon which when clicked hides the "torn off" window. The specific details of the individual panes are listed below.

Color

VIDA provides a wide array of coloring schemes as can be seen in the color pane (see *Figure: Color Pane*). The first button (with the color wheel icon) prompts the user to select a specific color which will be applied to all of the objects in the current scope. The second button (with the molecule overlaid on a color wheel) prompts the user to select a specific color that will be applied to all molecules in the current scope. In addition, this button contains a drop down menu of molecule specific coloring schemes which can be applied. Detailed information about the individual color schemes can be found in *Molecule Color*. The third button (with the surface overlaid on a color wheel) prompts the user to select a specific color that will be applied to all surfaces in the current scope. In addition, this button contains a drop down menu of surface specific coloring schemes which can also be applied. Detailed information about the individual color schemes can be found in *Surface Color*. The fourth button (with the grid overlaid on a color wheel) prompts the user to select a color that will be applied to all grid contours in the current scope. The fifth button (with the "U" icon overlaid on a color wheel) assigns a unique color to every object in the current scope. The last button restores the original color of all the objects in the current scope. More details about the specific coloring schemes can be found below.

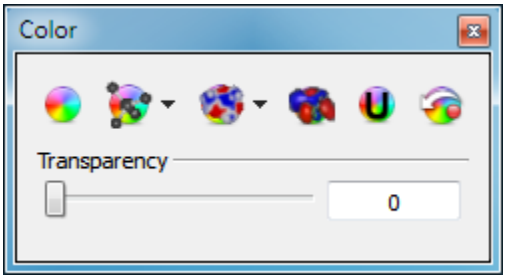
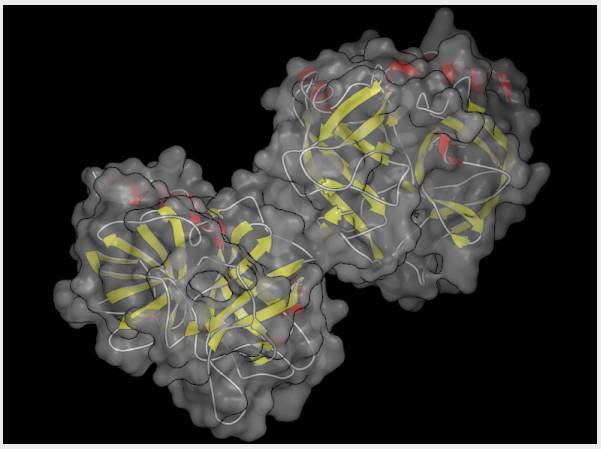
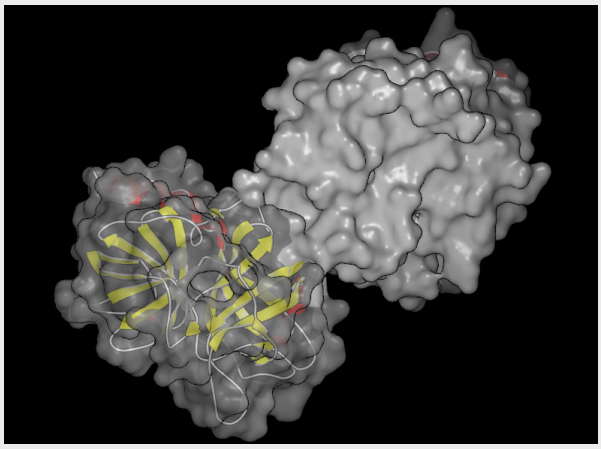


Fig. 2: **Color Pane**

Beneath the buttons is a slider which can be used to adjust the transparency of surfaces (see *Table: Surface Transparency*) and grid contours. The slider values range from 0 (completely opaque) to 100 (completely transparent). The specified transparency value is applied to all surfaces and grid contours in the current scope. However, if a part of a surface is selected, the transparency will only be applied to the selected region as can be seen in the second figure in *Table: Surface Transparency*.

Table 1: Surface Transparency

	
Transparent surface	Partially transparent surface

Selection

The simplest mechanism to perform a selection is to click on the object of interest in the 3D display using the mouse. However, a number of additional selection mechanisms are available in the selection pane as can be seen in *Figure: Selection Pane*.

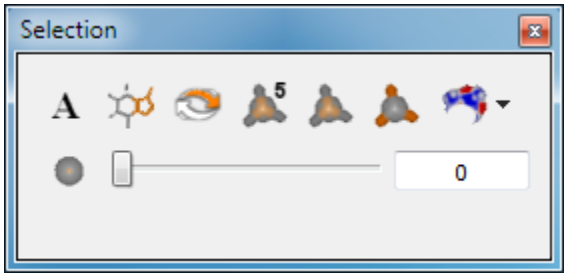


Fig. 3: **Selection Pane**

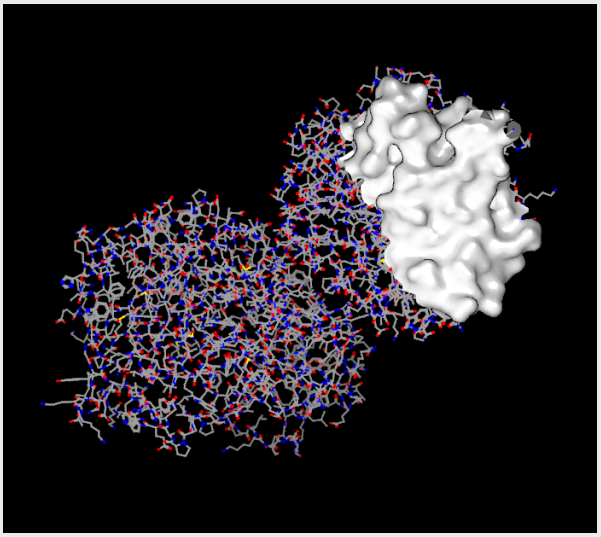
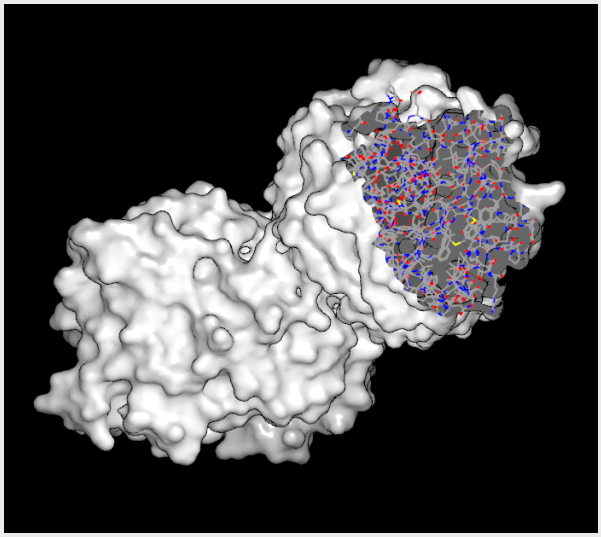
In the top row of the selection pane, there are seven individual buttons. The first button with the "A" icon selects all of the currently visible molecules. The second button with the partially selected molecule icon performs a selection based on a substructure query. Clicking on this button will launch a query dialog which allows the user to specify a substructure query. The substructure can be entered as a SMARTS pattern, selected from a large number of predefined patterns, or can be specified using a common or IUPAC name which will be converted to a structure using OpenEye's **Lexichem** toolkit. All atoms in the current scope matching the specified pattern will be selected. If the current scope is set to *All* and more than 50 molecules match the pattern, the user will be prompted as to whether to *Mark* the matching structures instead of selecting them, as being selected would make them visible.

The third button inverts the current selection, unselecting everything that was selected and selecting everything that is currently visible but was not previously selected. The fourth button selects everything within a defined radius of the current selected set (the default is 5 Angstroms). The fifth button selects everything within a user specified radius of the current selected set. This radius is specified by the slider in the bottom row of the pane. The sixth button selects everything outside of the same user specified radius of the current selected set. It is important to note that these distance based selection mechanisms expand their selected sets to include entire residues in the event that only a portion of a residue is included within the radius.

The last button contains a drop down menu which has different options based on the current selection. If a molecule is selected it contains the following options: "Hide Outside", "Hide Inside", and "Restore". Selecting one of these

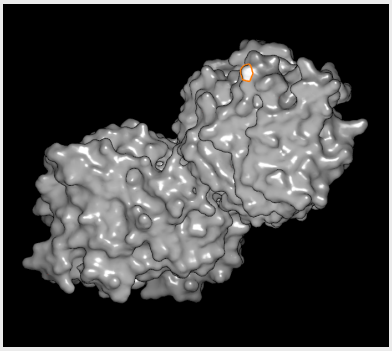
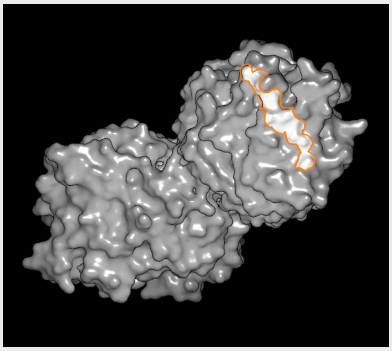
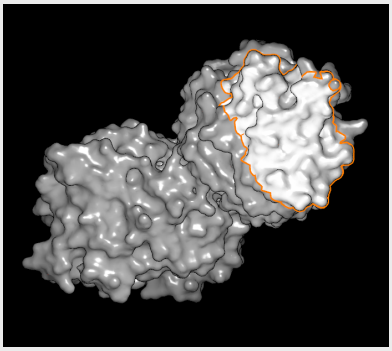
options will hide all of the atoms and bonds outside the current slider specified radius, hide all of the atoms and bonds inside the current slider specified radius, or unhide any previously hidden atoms and bonds respectively. If a surface is the options are: “Crop Outside”, “Crop Inside”, and “Restore”. Selecting one of these options will crop away the unselected portion of the surface, crop away the selected portion of the surface, or restore the surface to its original state respectively. Examples of cropped surfaces can be seen in *Table: Surface Cropping*.

Table 2: Surface Cropping

	
Unselected region removed	Selected region removed

In the bottom row, there is one button adjacent to a slider and a numeric display. The button is a toggle that controls whether or not unselected atoms and bonds are shown based on their distance to the selected set. The cutoff distance is controlled by the adjacent slider (and the actual value in Angstroms can be seen in the numeric display). This state of this button is ignored when a surface is selected. When a vertex (or a line of vertices) is selected, the slider can be used to flood out from the original selected set to include many more adjacent vertices. The different types of surface selections can be seen in *Table: Surface Scribing*.

Table 3: Surface Scribing

		
Single atom selected	Line of atoms selected	Region of surface selected

Style

The style pane of the Style Control is divided into two distinct areas as can be seen in *Figure: Style Pane*. The area above the dividing line contains a row of four buttons which control the display style of the various different types of objects that can be visualized in VIDA. The area beneath the dividing line is a collection of buttons whose functionality is specific to just molecules.

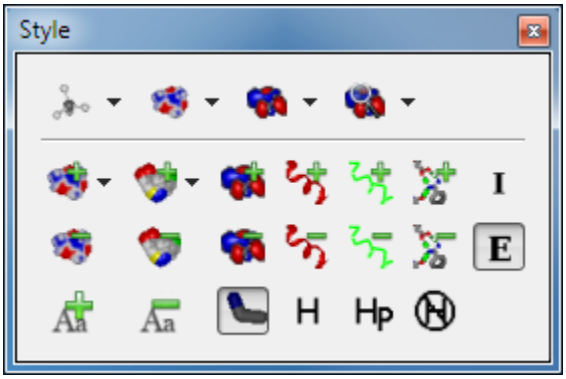


Fig. 4: **Style Pane**

In the top area, the first button (with the molecule icon) contains a drop down menu of the available molecule display styles: *ball and stick*, *CPK*, *stars*, *stick*, *wireframe*, and *hidden*. Selecting one of these options will change the display style of all the atoms and bonds in the current scope.

The second button (with the surface icon) contains a drop down menu of the available surface display styles: *solid*, *mesh*, and *points*. Selecting one of these options will change the display style of all the surfaces in the current scope.

The third button (with the grid icon) contains a drop down menu of the available grid display styles: *solid*, *line*, and *cloud*. Selecting one of these options will change the display style of all the grid contours in the current scope.

The fourth button contains a drop down menu of the supported grid types in VIDA: *Electrostatic*, *ET*, *FRED*, *Generic*, *Difference Map*, and *Regular Map*. Selecting one of these options will change the grid type of all the grids in the current scope. A grid's type determines how it is visualized in VIDA including the default number of contours (and their levels) and the display style as well as the color of those contours.

In the bottom area, there are three individual rows of buttons. In the first row, the first three buttons turn on the display of molecular surfaces, accessible surfaces, and electrostatic grids, respectively. Surfaces are created over the selected atoms, or over the currently scoped molecule if no atoms are selected. The two surface buttons have pulldown menus to allow either toggling the visibility of a surface or regeneration of the surface. Regeneration may be used to surface a new part of a molecule if the selection has changed. Surfaces and grids generated with these buttons are considered display properties of their associated molecules and therefore are not displayed in the List Window. Independent (non-property) versions of these can be created from the relevant options in a right-click menu. The next two buttons turn on the display of protein ribbons and c-alpha traces respectively. The next button turns on the display of hydrogen bonds by making all of the molecules in the current scope to be hydrogen bond targets. Being a hydrogen bond target means that when it is visible, it displays any hydrogen bonds made between it and any of the other molecules that are also visible. The last button in the row toggles whether or not internal (intramolecular) hydrogen bonds are shown.

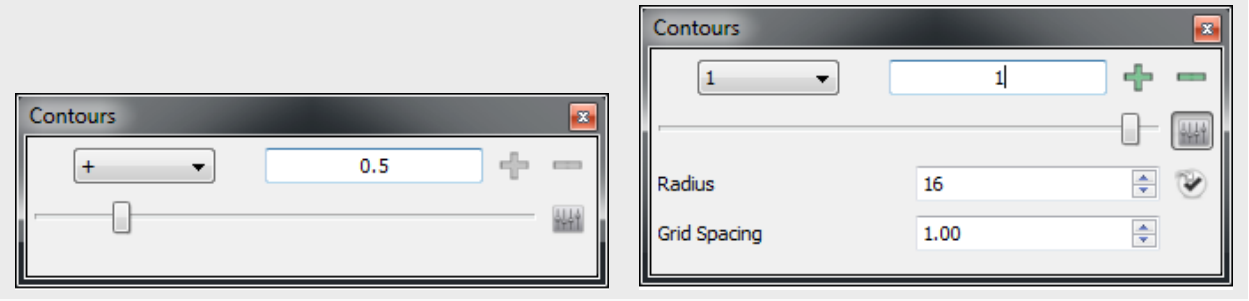
In the second row, the first three buttons turn off the display of molecular surface, accessible surfaces, and electrostatic grids, respectively. The next two buttons turn off the display of protein ribbons and c-alpha traces. The next button makes all of the molecules in the current scope no longer hydrogen bond targets. The last button in the row toggles whether or not external (intermolecular) hydrogen bonds are shown.

In the last row, the first button allows the user to specify and turn on atom and bond labels. The second button turns off all of the atom and bond labels in the current scope. The third button toggles the display of non-bonded atoms in the

scene. The remaining three buttons control the hydrogen style for the molecules in the current scope. The first of the three turns all hydrogens on, the second buttons shows only polar hydrogens, and the last button hides all hydrogens.

Contours

Table 4: Contour Pane



Normal view *Advanced view*

The contour pane of the Style Control is used to control the number and level of the individual grid contours as can be seen in *Table: Contour Pane*. When the current application scope is set to *Focused* there will be a pull down menu in the top left of the window containing a list of all the current grid contours. If the scope is not set to *Focused*, the scope will be displayed in place of the pull down menu to remind users that all operations in this window apply to more than just the *Focused* grid. Next to this area is a numeric display which displays the current contour level. Next to this display are two buttons, one with a "+" icon and one with a "-" icon. These two buttons allow for the creation and deletion of individual grid contours. Please note that these options are disabled for both *Electrostatic* and *ET* grids as they can have only two contours: positive and negative. Furthermore, the contour levels of the positive and negative contours are coupled together and so changing one will change the other (but with the opposite sign).

At the bottom of this window is a slider which allows for direct control over the contour level. To the right of the slider is an *Advanced* button (with the equalizer icon) which will expand the pane to show a number of advanced options (see *Table: Contour Pane*).

In the advanced section, there are two adjustable parameters (*Radius* and *Grid Spacing*) and one button. The parameters control how contours of *Reentrant* grids are displayed according to their symmetry. The *Radius* parameter specifies how far out from the center of the scene (in Angstroms) VIDA should look in the grid in order to generate of the isocontour. The *Grid Spacing* parameter specifies the spacing (in Angstroms) of the sub-grid used to generate the isocontours.

The single button (with the check icon) to the right of the *Radius* parameter converts the current grid contour into a fixed surface independent of the grid. This can be particularly useful when generating state files as a single surface can take up considerably less disk space than an entire grid.

Graphics

The graphics control pane in the Style Control provides an additional level of control over and interaction with the 3D display (see *Figure: Graphics Pane*). It is particularly useful for users without mouse wheels.

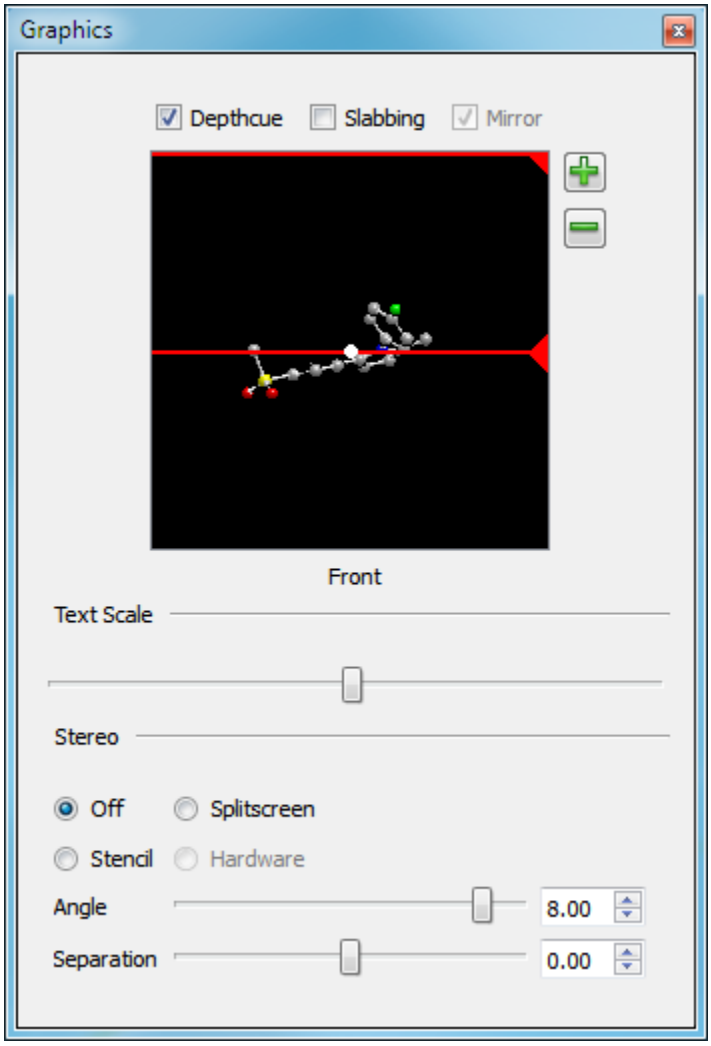


Fig. 5: **Graphics Pane**

At the top of the graphics pane are three check boxes: “Depthcue”, “Slabbing”, and “Mirror”. The first two check boxes toggle the use of their associated properties respectively. The “Mirror” check box is only enabled when slabbing is on and controls whether the front and back slabbing planes move together (mirrored around the center of the scene) or independently.

Beneath these check boxes is an interactive control which allows the user easy control over both depthcueing and the clipping planes. Displayed in the control is a top-down view of the current scene in the 3D display. The front of the scene is at the bottom as indicated by the label.

When depthcueing is enabled, two red lines will appear in the control which indicate the start and the end points in the depthcueing calculation. These lines can be moved by clicking on the triangular tab at the right of the lines and dragging them to the desired location. The 3D display will update dynamically as these lines are moved.

When clipping (or slabbing) is enabled, two yellow lines will appear in the control which indicate the position of the near and far clipping planes. These can be moved by simply clicking on the triangular tab at the left of the lines and dragging them to the desired location. The 3D display will update dynamically as these lines are moved.

In the center of the control is a small white dot that corresponds to the center of scene that the camera is pointing at. The position of the center can be changed in the Z plane by dragging the dot in the vertical plane. The position of the center can be changed in the X plane by dragging the dot in the horizontal plane.

Beneath this control is a slider which controls the scale of the text drawn in the 3D display. The 3D display will update dynamically as the slider is adjusted.

Beneath the slider are four radio buttons which control the mode of stereoscopic visualization. The default behavior is "Off". The "Stencil" option provides an additional mechanism for hardware stereo specific to the Zalman Trimon 3D LCD Monitor.

Custom Views

Each of the individual custom views available in VIDA can also provide their own individual panes which can be inserted into the Style Control to further enable task specific operations. These panes appear when a specific custom view is selected and disappear when the "X" icon is clicked or when another custom view is loaded. For more information on custom views refer to the *Custom Views* chapter.

4.2 Rendering

The images shown in the 3D display are rendered using OpenGL, therefore the quality and speed of the rendered scenes in the 3D display is dependent on the graphics card in the computer running the application. Computers with graphics cards that do not support OpenGL acceleration will still be able to display images, but the quality and speed with which the images are displayed may be markedly reduced. To help alleviate situations like this, multiple "levels of detail" are supported ranging from "Fastest" to "Best" which adjust the quality of the rendered scenes accordingly. For most computers, "Best" level of detail should be appropriate. "Presentation" quality rendering is optimized for creating images with a white background, such as might be used in slide presentations and printed documents.

In addition to the "level of detail", the scene can be further adjusted by modifying a few select OpenGL parameters such as the background color as well as the lighting and material models. All of these parameters can be adjusted in the *application preferences*.

4.3 Stereo

Multiple types of stereo visualization are supported in the 3D display. The "Stencil" option enables stereo visualization using the Zalman Trimon 3D LCD Monitor. This is the only LCD display on which stereo visualization is supported. This monitor has been tested on multiple operating systems and requires no specific graphics card.

Finally, split-screen stereo is also a viable option. Both cross-eyed and wall-eyed views are available and can be set in the *preferences*. Furthermore, the stereo angle and eye offset parameters can be adjusted in the *preferences* as well.

4.4 Viewpoint

When viewing objects in the 3D display, there is always a center to the scene where the virtual "camera" is pointing. When the first molecule, grid, or surface is loaded into the application, the center of that object becomes the center of the scene where the camera is pointing. By default, the center does not automatically change with changes to the scene. The center can be changed explicitly from the main application toolbar, the right-click popup menu, or using the following scripting commands: `ViewerCenterSet` or `ViewerCenterAndRadiusSet`. Furthermore, the center can also be translated using mouse interactions.

The center can be made to automatically change when browsing if the "Auto-Center" or "Auto-Fit" preferences are set to true (the default is false). The "Auto-Center" preference directs the window to always recenter the scene after any changes are made to that scene. The "Auto-Fit" preference directs the window to not only recenter the scene after any changes are made but to also ensure that everything in the scene is completely visible in the window. These preferences can be edited in the *application preferences* dialog.

4.5 Bookmarks

The 3D display is capable of capturing and storing named bookmarks of the currently displayed scene to enable viewing again at another time without having to setup the scene again by hand. Bookmarks are saved in state files and are particularly useful when trying to communicate multiple pieces of information to other users.

All bookmarks have names and are stored in order. Bookmarks can be created from the top-level *Bookmark* menu by selecting the "Add" option. The order of the bookmarks can be organized by selecting the "Organize" option in the same menu. The *Bookmark* menu also contains an "Animated" option which controls whether VIDA animates the transition between the current scene and the newly selected bookmark. A list of all the currently available bookmarks populate the rest of the menu. Bookmarks can also be created by clicking on the "Add Bookmark" button in the 3D display *Style* toolbar.

The 3D display also provides a *bookmark display widget* which displays a clickable list of all the available bookmarks directly in the 3D display.

4.6 Tiled Display

The default behavior of the 3D display is to display everything that is *Visible* in the same coordinate space; however, there are often occasions when it is desirable to view everything in their own space. This behavior can be obtained in "Tiled" mode, also called "Matrix" mode or "Multi-Pane" mode. "Tiled" mode can be toggled on or off from the main application toolbar.

When the 3D display is in "Tiled" mode every *Visible* and *Focused* object is displayed centered in its own individual pane. However, if desired each object can be shown relative to the same center if the "Auto-Center Panes" option in the *3D Display* section of the application preferences is unchecked. It is important to note that when "Auto-Center Panes" is enabled it is not possible to recenter or translate using mouse interactions in any of the panes.

To indicate which object is *Focused*, the corresponding pane will contain a blue border just inside the regular pane border. If there are any *Locked* objects, each of them will be displayed in every pane in addition to the individual *Visible* and *Focused* objects.

4.7 Toolbars

The 3D display provides additional toolbar areas (aside from the main application toolbar) inside the 3D window. There are four possible toolbar locations: at the top of the window, at the bottom of the window, at the left edge of the window and at the right edge of the window.

By default, the 3D display provides a "Mouse" toolbar at the left edge of the display. In Edit mode, a "3D Right" toolbar on the right side of the display is populated with building tools. Two toolbars are empty and hidden by default, one at the bottom of the 3D Window ("3D Bottom") and one at the top ("Style"). These can be shown and populated *via* the use of scripting commands.

4.7.1 Mouse Toolbar

The *Mouse* toolbar contains a single row of buttons along the left side of the 3D display which control the behavior of the mouse in the display. The first button is the default and puts the mouse into normal selection mode. The second button puts the mouse into information mode which will cause information about the object currently under the mouse to be displayed. The third button puts the mouse into centering mode which will cause the display to be centered on whatever the mouse clicks on. The fourth button puts the mouse into measuring mode. There is a drop down menu adjacent to this button which allows the user to select one of three measurement modes: *Distance*, *Angle*, and *Torsion*. The fifth button puts the mouse into labeling mode which will prompt the user for a custom label to apply to any object that is clicked.

Double-clicking in the background of the 3D display will restore the mouse function to the default selection mode.

This toolbar can be referenced in scripting commands using the name “Mouse”.

4.7.2 Bottom Toolbar

Although empty by default it contains a set of buttons (that change depending on what is active) collectively referred to as the “Object toolbar” although they are not technically a separate toolbar. It also contains the *bookmark widget*.

For each type (molecule, grid, surface) it can show two buttons which represent menus - one color & one style. If the active object is a molecule and it has surfaces and/or grids shown via the Style Control, it will show the button menus for those objects as well. These buttons are quick links to the options available in the *Style Control* window.

This toolbar can be referenced in scripting commands using the name “Bottom”.

4.7.3 Right Toolbar

This toolbar is displayed during 3D building, and contains tools for editing molecules. The first button is a tool for deleting atoms and bonds. The second button shows a benzene ring by default and attaches the fragment selected from a dropdown to either an atom or bond of the molecule in the main window. The third button provides a dropdown for selection of the active atom type. The fourth button provides a selection of bond types from a dropdown. The fifth button allows modification of format atomic charges. The sixth button allows stereochemistry to be set (R/S or E/Z) or inverted. The last tool on this toolbar minimizes the active molecule using the MMFF94 or MMFF94s forcefield.

4.7.4 Application Toolbar

In addition to providing its own toolbars, the 3D display is affected by a few of the buttons in the main application toolbar. The application toolbar contains the “Tiled” display toggle button (matrix icon), the main window screenshot button (camera icon), the centering button (four arrows pointing in), the fitting button (four arrows pointing out), the new molecule button, and the edit molecule button.

The “Tiled” display button toggles the display between single pane overlay mode and multiple pane tiled mode. The screenshot button captures an image of the current main window at a user specified resolution. The centering button centers the display on the currently selected set or whatever molecule(s) are in the current scope. The fit button centers the scene and adjusts the scale factor such that all *Visible* objects can be seen on the screen at the same time. The new molecule button prompts the user to specify a new molecule using the 2D molecule editor. The edit molecule button puts the application into editing mode for the currently active molecule.

4.8 Display Widgets

The 3D display provides a number of additional widgets which are drawn directly in 3D display as opposed to on the side or floating above it. These widgets include an annotation widget, a bookmark widget, a data display widget, and a depiction widget. See *Figure: 3D Display Widgets*.

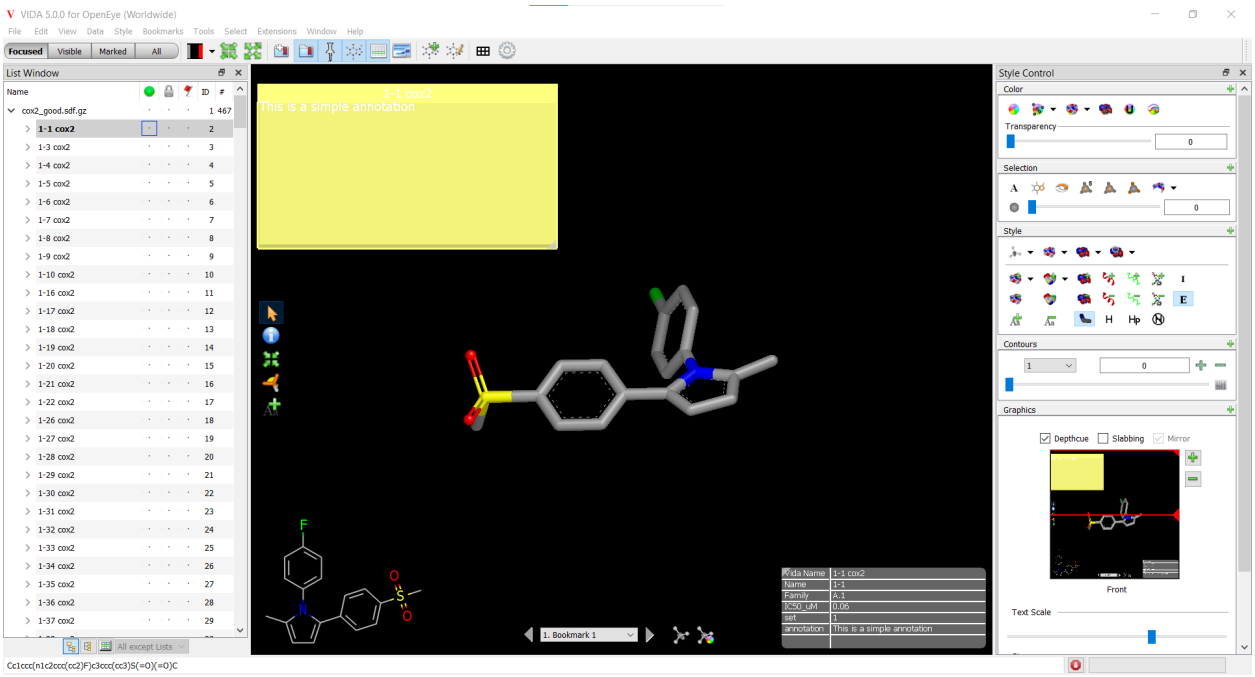


Fig. 6: **3D Display Widgets**

4.8.1 Annotation

It is often desirable when viewing molecules to make notes or simple annotations about specific molecules. This can be done directly within the 3D display, where the annotation widget appears like a transparent Post-It(TM) (see *Figure: Molecule Annotation*). The color of the widget as well as the font used can be specified in the *application preferences*.

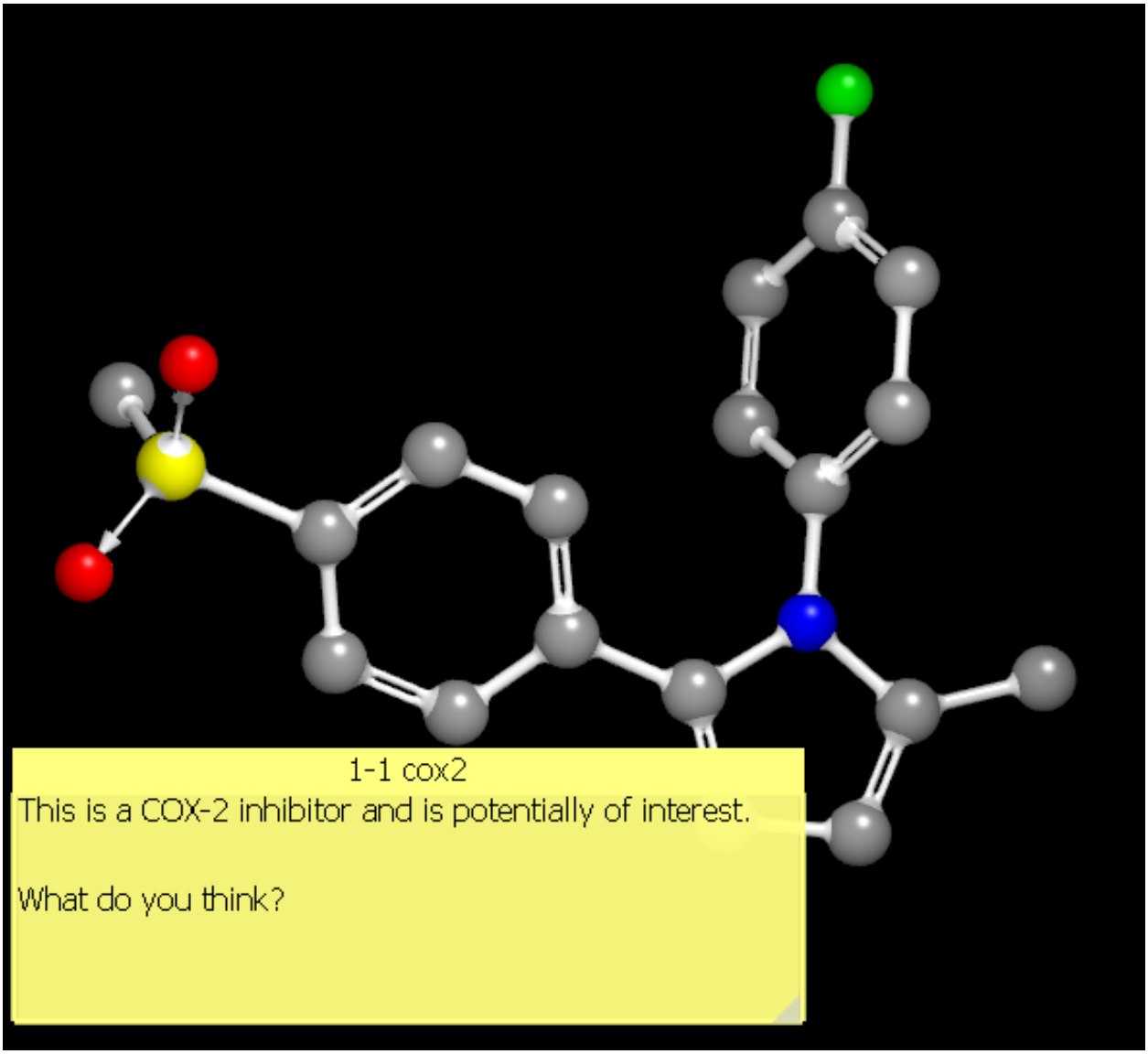


Fig. 7: **Molecule Annotation**

The widget can be moved by clicking on the title bar and dragging it to the desired location. The widget can be resized by grabbing the small triangular tab in the lower right-hand corner of the widget and dragging until the desired size is achieved.

Clicking in the annotation widget will activate it for editing. Editing can be accomplished by directly typing once the widget has been clicked or by clicking on the "Edit" button in the title bar. Clicking on the "Edit" button will launch a more fully featured text editor which supports copy and paste operations (see *Figure: Annotation Editing*).

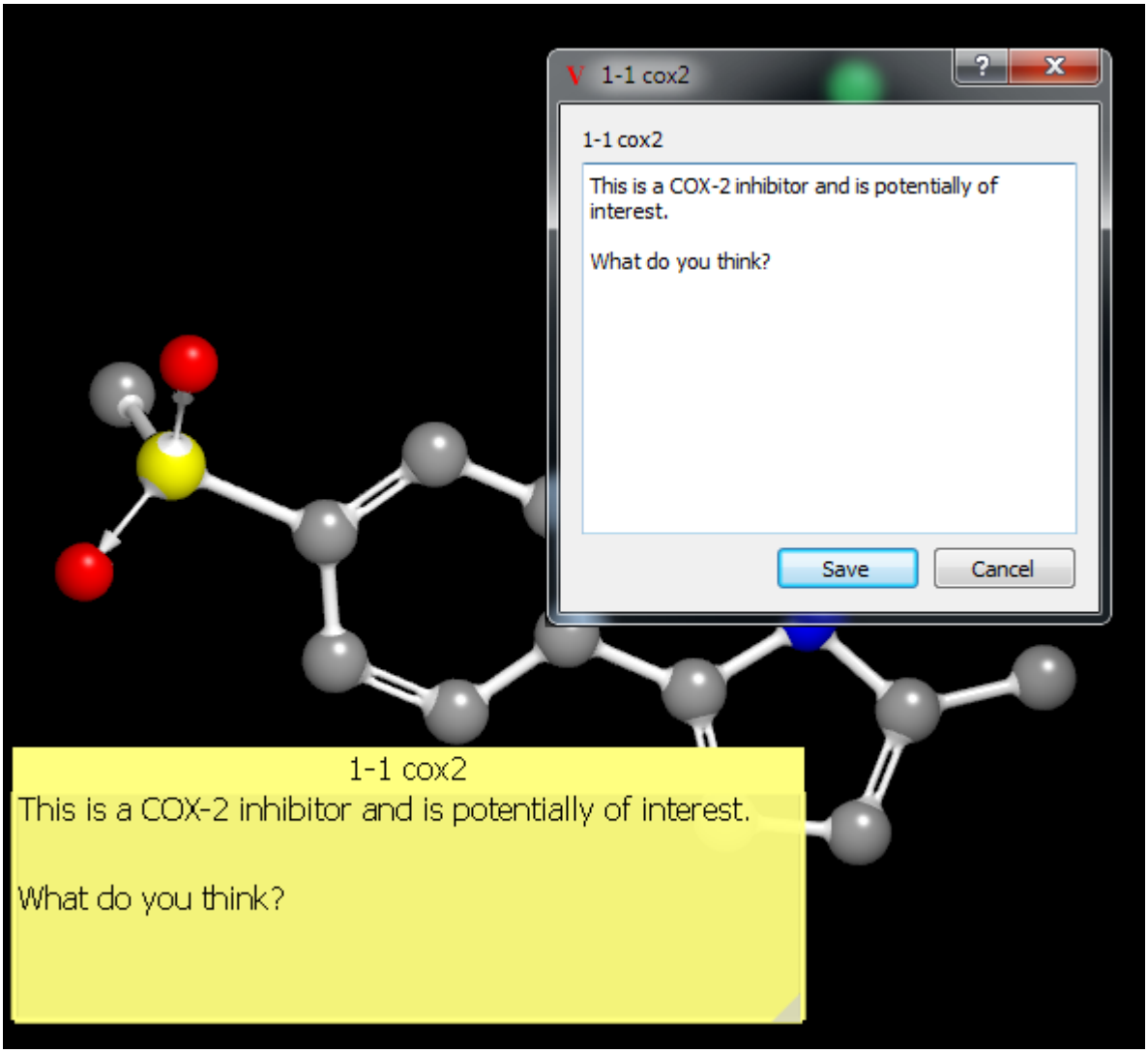


Fig. 8: *Annotation Editing*

Annotations are also stored in the spreadsheet and can be edited there as well as in the 3D display. Annotations are saved as a SD tag when a molecule is written out (in either SD or OEB formats).

4.8.2 Bookmark

The bookmark widget is a simple widget drawn at the bottom of the 3D display (*Figure: 3D Display Widgets*) containing a clickable list of the currently available bookmarks. Clicking on any individual bookmark will load that bookmark and update the display accordingly.

4.8.3 Data Display

This widget provides a simple table which contains the spreadsheet data of the *Focused* object. The widget can be moved around the 3D display by clicking and dragging it to the desired location. It can also be resized by clicking in the tab in the upper left hand corner. Lastly, if the amount of data associated with the object is too large to be shown all at once, the widget does provide scrolling functionality.

4.8.4 Depiction Widget

The depiction widget draws a 2D depiction of the focused molecule directly into the 3D display. The depiction is drawn in the lower left hand corner of the display (or individual pane if in tiled mode). The depiction style is determined based on the application 2D depiction preferences. The size and other drawing parameters (e.g. anti-aliasing) of the depiction are also specified in the *application preferences*.

The depiction widget is mouse-aware making it possible to perform selections in this region.

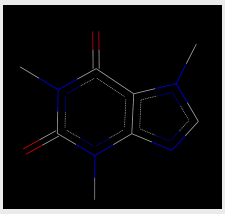
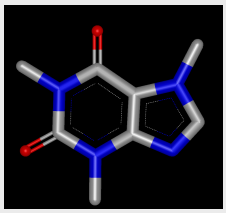
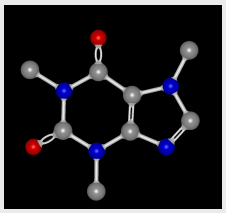
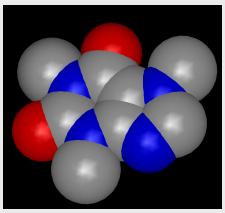
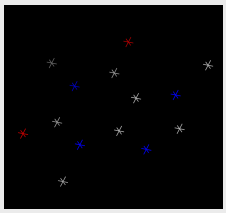
4.9 Molecular Visualization

This section describes the very powerful, versatile, and interactive molecular visualization capabilities of the 3D display. The interactive control of molecular visualization is typically performed using the *Style Control* or through scripting commands.

4.9.1 Display Styles

All of the standard molecule display styles are available in VIDA (*Wireframe*, *Stick*, *Ball and Stick*, *CPK*, and *Stars*), and are illustrated in *Table: Display Styles*. By default, small molecules are drawn in *Stick* mode while large molecules are drawn in *Wireframe* mode. These defaults can be changed in the *application preferences*.

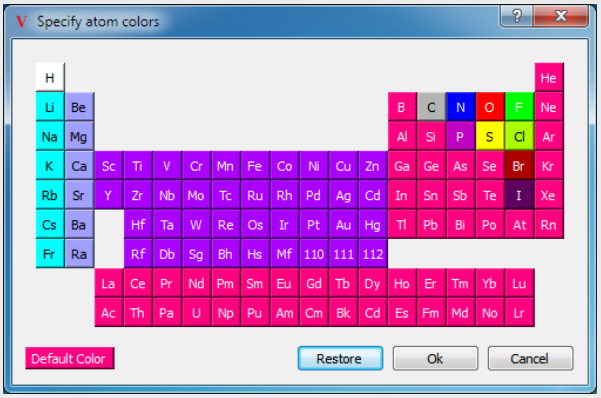
Table 5: Display Styles

				
Wireframe	Stick	Fancy Ball and Stick	CPK	Stars

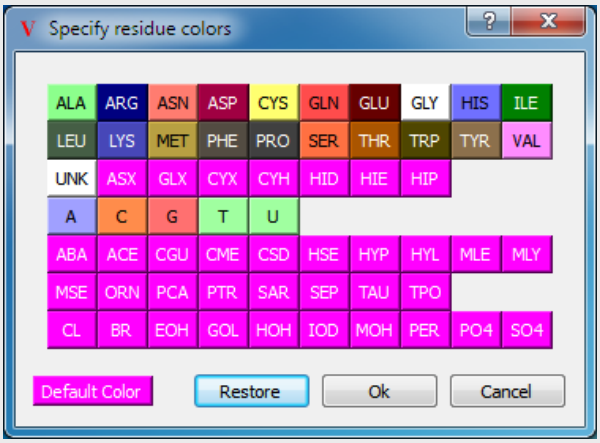
4.9.2 Molecule Color

By default, the atoms in a molecule are colored according to their element types and the bonds in a molecule are colored according to the element types of the two atoms defining that bond. There are two color palettes that specify the actual colors used for a given element, one for use with dark-colored backgrounds and the other for use with light-colored backgrounds. Both of these palettes can be viewed and modified in the application preferences. The dark-background atom color palette can be seen in *Table: Color Palettes*.

Table 6: Color Palettes



Atom Color Palette



Residue Color Palette

Coloring Schemes

There are a number of molecule specific coloring schemes in addition to the standard element based coloring. The following schemes are discussed below: amino, bfactor, carbon, chain, cpk, cpknew, element, formal charge, group, partial charge, reference, residue, and shapely.

Amino

The *amino* color scheme is a protein specific scheme which colors atoms according to their individual residues. The residue colors are listed below.

Table 7: Amino Color Scheme

Residue	Color	RGB Value
ASP, GLU	Bright Red	[230, 10, 10]
CYS, MET	Yellow	[230, 230, 0]
LYS, ARG	Blue	[20, 90, 255]
SER, THR	Orange	[250, 150, 0]
PHE, TYR	Mid Blue	[50, 50, 170]
ASN, GLN	Cyan	[0, 220, 200]
GLY	Light Grey	[235, 235, 235]
LEU, VAL, ILE	Green	[15, 130, 15]
ALA	Dark Grey	[200, 200, 200]
TRP	Purple	[180, 90, 180]
HIS	Pale Blue	[130, 130, 210]
PRO	Flesh	[220, 150, 130]
Others	Tan	[190, 160, 110]

BFactor

The *bfactor* color scheme is a protein specific scheme which colors atoms according to the bfactor value using a fixed blue-to-red gradient between 0 and 60.

Carbon

The *carbon* color scheme colors all of the carbon atoms in the current scope the same color which is specified by the user.

Chain

The *chain* color scheme is a macromolecule specific scheme which assigns a unique color to each of the macromolecular chains.

CPK, CPKNew

The *CPK* color scheme is an element specific scheme based on the colors used in the popular CPK plastic space filling models. The *CPKnew* color scheme is a variant on the *CPK* color in that it uses a slightly brighter version of some of the colors. The atom colors are listed below with the RGB values for *CPK* in the third column and the RGB values for *CPKNew* in the fourth column (if different).

Table 8: CPK, CPKNew

Element	Color	CPK: RGB Values	CPKNew: RGB Values
H	White	[255, 255, 255]	
He	Pink	[255, 192, 203]	
Li	Fire Brick	[178, 34, 34]	[178, 33, 33]
B,Cl	Green	[0, 255, 0]	
C	Light Grey	[200, 200, 200]	[211, 211, 211]
N	Sky Blue	[143, 143, 255]	[135, 206, 235]
O	Red	[255, 0, 0]	
F,Si,Au	Golden Rod	[218, 165, 32]	
Na	Blue	[0, 0, 255]	
Mg	Forest Green	[34, 139, 34]	
Al,Ca,Ti,Cr,Mn,Ag	Dark Grey	[128, 128, 144]	[105, 105, 105]
P,Fe,Ba	Orange	[255, 165, 0]	[255, 170, 0]
S	Yellow	[255, 200, 50]	[255, 255, 0]
Ni,Cu,Zn,Br	Brown	[165, 42, 42]	[128, 40, 40]
I	Purple	[160, 32, 240]	
Unknown	Deep Pink	[255, 20, 147]	[255, 22, 145]

Element

The *element* color scheme is the default color scheme used when coloring molecules. Colors are assigned according to atomic number. The actual colors used for a given element can be edited in the *application preferences* as seen in *Table: Element Color Palettes*.

Table 9: Element Color Palettes

Element	Color	RGB Values
H	White	[255, 255, 255]
C	Grey	[180, 180, 180]
N	Blue	[0, 0, 255]
O	Red	[255, 0, 255]
F	Green	[0, 255, 0]
P	Magenta	[192, 0, 192]
S	Yellow	[255, 255, 0]
Cl	Lime Green	[170, 255, 0]
Br	Dark Red	[170, 0, 0]
I	Dark Orange	[170, 85, 0]
Group I	Cyan	[0, 255, 255]
Group II	Light Blue	[160, 160, 255]
Transition Metals	Purple	[170, 0, 255]
Others	Pink	[255, 0, 128]

Formal Charge

The *formal charge* color scheme colors atoms according to their formal charge using a fixed red-to-blue gradient between -4 and +4.

Group

The *group* color scheme is a macromolecule specific scheme which colors each atom according to its position in a macromolecular chain. The colors are assigned along a smooth rainbow spectrum from blue, through green, yellow and orange to red.

Partial Charge

The *partial charge* color scheme colors atoms according to their partial charge using a red-to-blue gradient between -1.0 and +1.0.

Reference

The *reference* color scheme colors all of the carbon atoms in the current scope using the current *Reference* color which can be set in the *application preferences*. The default is green.

Residue

The *residue* color scheme is a protein specific scheme which colors atoms according to their individual residues. The actual colors used can be edited in the *application preferences* as seen in *Table: Color Palettes*. The default colors correspond to those used in the *shapely* color scheme.

Shapely

The *shapely* color scheme is a protein specific scheme which colors atoms according to their individual residues. This scheme is based upon Bob Fletterick's "Shapely Models"([Fletterick-1982]). The residue colors are listed below.

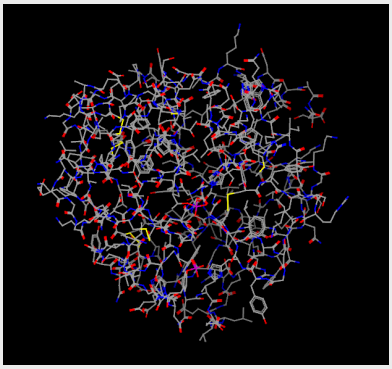
Table 10: Shapely Color Scheme

Residue	Color	RGB Values
ALA	Medium Green	[140, 255, 140]
GLY	White	[255, 255, 255]
LEU	Olive Green	[69, 94, 69]
SER	Medium Orange	[255, 112, 66]
VAL	Light Purple	[255, 140, 255]
THR	Dark Orange	[184, 76, 0]
LYS	Royal Blue	[71, 71, 184]
ASP	Dark Rose	[160, 0, 66]
ILE	Dark Green	[0, 76, 0]
ASN	Light Salmon	[255, 124, 112]
GLU	Dark Brown	[102, 0, 0]
PRO	Dark Grey	[82, 82, 82]
ARG	Dark Blue	[0, 0, 124]
PHE	Olive Grey	[83, 76, 66]
GLN	Dark Salmon	[255, 76, 76]
TYR	Medium Brown	[140, 112, 76]
HIS	Medium Blue	[112, 112, 255]
CYS	Medium Yellow	[255, 255, 112]
MET	Light Brown	[184, 160, 66]
TRP	Olive Brown	[79, 70, 0]
ASX,GLX,PCA,HYP	Medium Purple	[255, 0, 255]
A	Light Blue	[160, 160, 255]
C	Light Orange	[255, 140, 75]
G	Medium Salmon	[255, 112, 112]
T	Light Green	[160, 255, 160]
Backbone	Light Grey	[184, 184, 184]
Special	Dark Purple	[94, 0, 94]
Default	Medium Purple	[255, 0, 255]

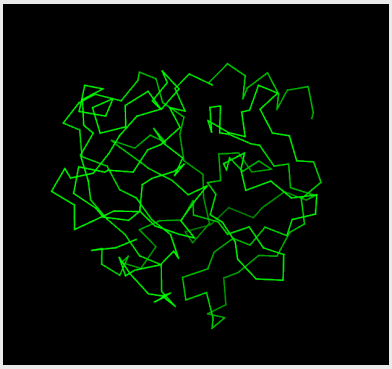
4.9.3 Proteins

In addition to the standard display styles, the two protein specific displays of *C-Alpha Traces* and *Ribbons* are supported. Examples of these display styles can be seen in *Table: Protein Styles*.

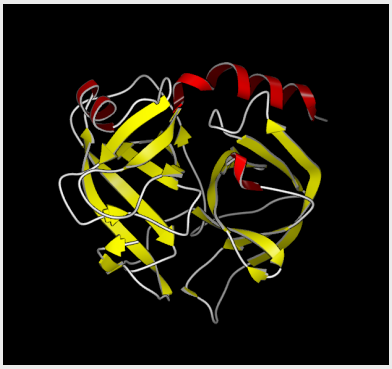
Table 11: Protein Styles



Wireframe



C-Alpha Trace



Ribbon

Given that protein files often inevitably contain many extra components that are not necessarily of interest to the user (waters for example), the ability to hide the display of non-bonded atoms is provided to ease the viewing of proteins without having to manually edit the input file. An example of a protein displayed with and without waters can be seen in *Table: Water Styles*. There is also a button in the *style panel* of the Style Control window that will toggle the display of non-bonded atoms.

Table 12: Water Styles

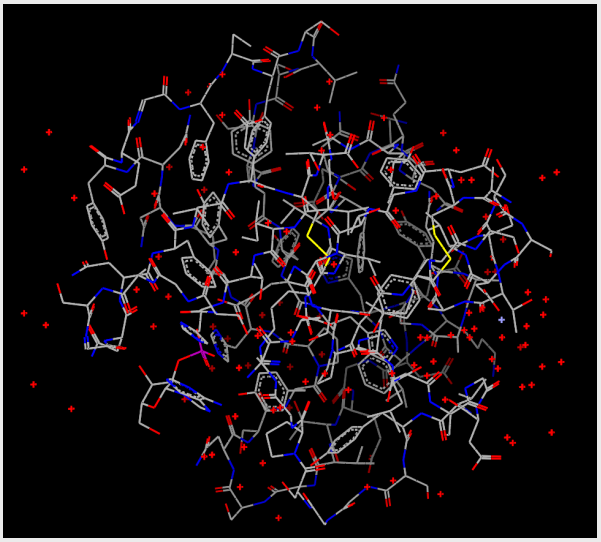
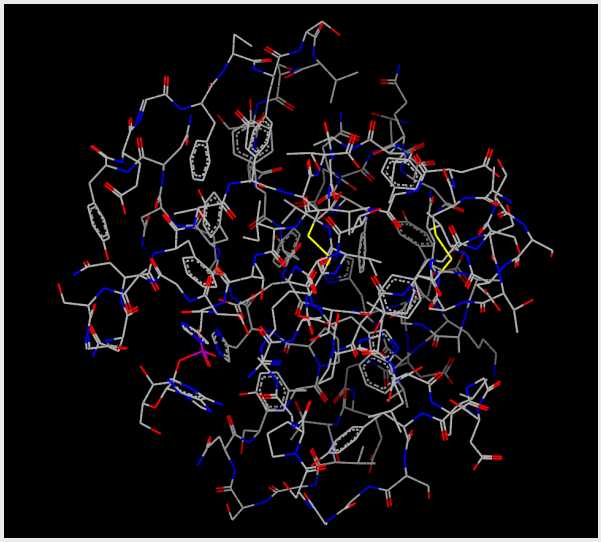
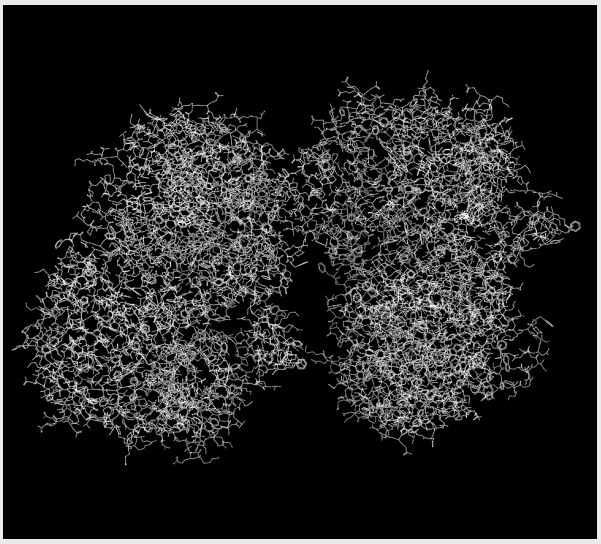
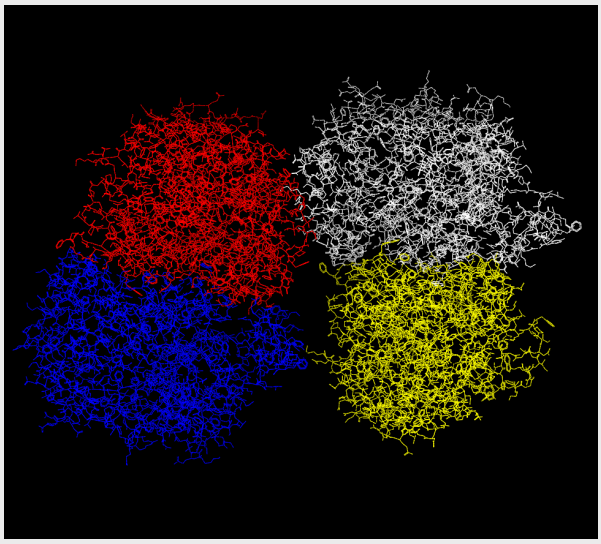
	
Protein with waters	Protein without waters

Table 13: Multimeric Styles

	
Multimer displayed as a single molecule	Multimer displayed as multiple molecules

Another common scenario when working with proteins is that the input file contains a large multimer which, as expected, is interpreted as a single molecule. However, sometimes the ability to interact with the individual components is of particular value. The ability to split a molecule into separate components can be performed by selecting the "*File* > *New Molecule* > *From Split* > Components" menu option. This function will then determine what the individual components of the *Focused* molecule are and create a new list containing a new molecule for each component. The

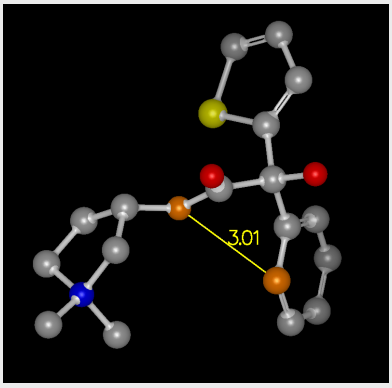
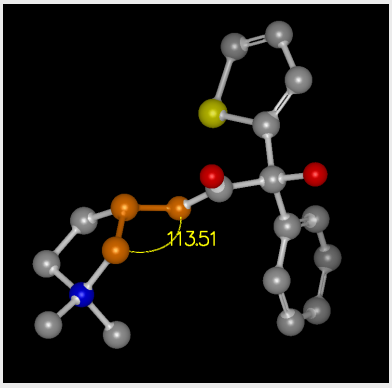
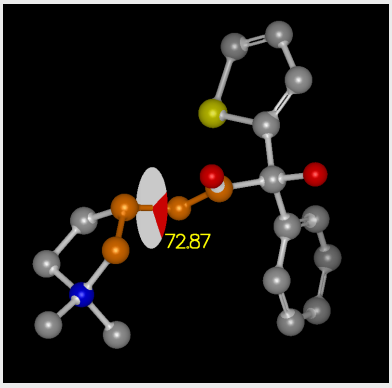
original molecule is not changed or deleted. An example of this process can be seen in Figure X.Y. The *From Split* submenu also contains "Selected" and "Marked" options which will create new molecules from the original based on the selected or marked set of atoms.

It may also be of interest to put some of these components back together into a single molecule. This can be done by choosing one of the following options from the "*File* > *New Molecule* > *From Merge*" submenu: "Selected", "Marked", "Visible". Choosing any one of these options will create a new molecule which is composed of all the molecules that were either "Selected", "Marked", or "Visible" at the time of the operation.

4.9.4 Monitors

The ability to display measurement monitors is a useful feature and is well supported. Access to these measurement facilities is available from the *Mouse* toolbar on the left hand side of the 3D display. There is a single button with three options which turn on *Distance*, *Angle*, and *Torsion* measurement respectively.

Table 14: Monitor Types

		
Distance Monitor	*Angle Monitor*	*Torsion Monitor*

When the mouse is put into one of these modes, the measurements are made based on the selected set of atoms. In *Distance* mode, the first atom selected is consider the "anchor" atom. Once the anchor atom has been selected, a temporary distance monitor will be displayed to any other atom that the mouse passes over. Selecting another atom will create a permanent monitor and clear the anchor. In *Angle* mode, the first two atoms selected are considered the anchor atoms and a temporary angle monitor will be displayed to any other atom that the mouse passes over. Selecting a third atom will create a permanent monitor and clear the anchors. In *Torsion* mode, the first three atoms selected are considered the anchor atoms and a temporary torsion monitor will be displayed to any other atom that the mouse passes over. Selecting a fourth atom will create a permanent monitor and clear the anchor.

Monitors can be removed by selecting the monitor of interest and then choosing the "Delete" option in the right-click menu. All of the visible monitors can be deleted in one step by right-clicking in the 3D display and selecting the "Delete visible monitors" option. Monitors are also displayed in the "List Window" and can be deleted individually there. Examples of the three types of monitors can be seen in *Table: Monitor Types*.

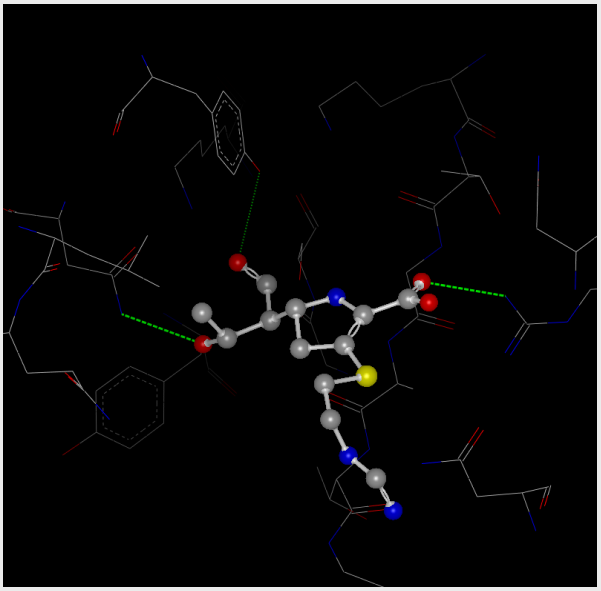
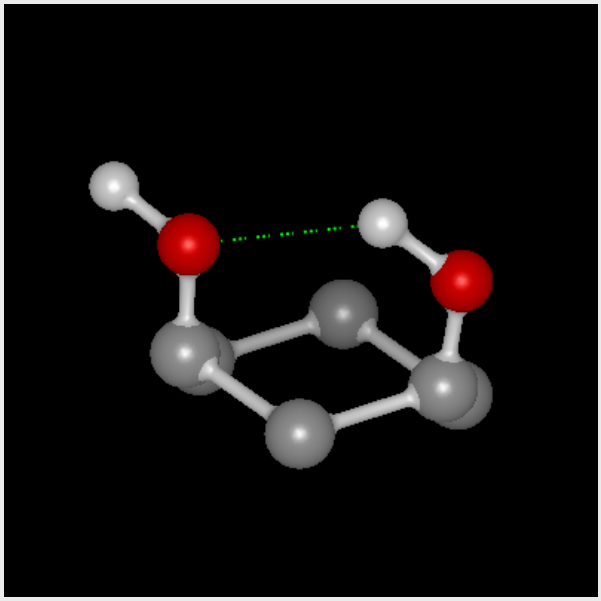
Monitors serve an important role when in editing mode as the properties they measure can be modified by selecting the monitor and then using the mouse wheel to adjust the value up or down. By default, the smaller portion of the associated molecule will be the part that is moved. This can be reversed by holding down the *Ctrl* key while using the mouse wheel. To achieve fine control of the values, hold down the *Shift* key while using the mouse wheel.

4.9.5 Hydrogen Bonds

Both internal (intramolecular) and external (intermolecular) hydrogen bonds can be displayed. Hydrogen bonds are determined by typing all of the atoms as either acceptors, donors, both, or neither. Then the hydrogen bond energy is calculated between all acceptors and donors based on the ChemScore scoring function which takes into account both

interatomic distance as well as the coordination geometry ([Eldridge-1997]). This function provides a relative score between 0 and 1.0 for the energy of the hydrogen bond. For all hydrogen bonds with an energy greater than 0, the bond is drawn as a dotted green line. The thickness of the line as well as the spacing between the dots correlates to the bond energy. Therefore, thicker more solid lines have higher energy than thinner more dotted lines. An example of both internal and external hydrogen bonds can be seen in *Table: Hydrogen Bonds*.

Table 15: Hydrogen Bonds

	
External Hydrogen Bonds	Internal Hydrogen Bonds

4.10 Grid Visualization

Grids are spatial arrays of data that are sampled at regular intervals. Grids may contain electrostatic samples of spatial regions, density samples, and other potential fields.

A grid can have a variable number of contours. Each contour has an index starting at 1 for the first contour. Additionally, every contour has its own individual color and a threshold value which can be positive or negative (also known as the "contour level"). There are three different display styles for grids contours: *Solid*, *Mesh* and *Cloud*.

4.11 Grid Types

As mentioned in the previous section, there are multiple different predefined grid types that are supported including: *Electrostatic*, *ET*, *FRED*, *Generic*, *Difference Map*, and *Regular Map*.

4.11.1 Electrostatic

Electrostatic grids are the only type of grid that can be actually be generated within VIDA as opposed to simply being visualized. Electrostatic grids are created for individual molecules using OpenEye's **Zap** toolkit (for more information, please see the **Zap** toolkit documentation). The creation of the grid requires the presence of partial charges on the input molecule. If partial charges were already present on the molecule, those will be used in the calculation. However, if no partial charges were present (or the user decides to ignore them, which can be set in the *application preferences*), temporary partial charges will be calculated using MMFF94 or AM1-BCC ([Jakalian-2000], [Jakalian-2002]). If a molecule has greater than 50 atoms, MMFF94 charging will be done even if AM1-BCC was selected. However, for

proteins an alternate residue based charging model is available and can be selected in the preferences. This model assigns partial charges based on residue information as is shown below:

Residue	Residue Atom	Partial Charge
ASP	OD1, OD2	-0.5
GLU	OE1, OE2	-0.5
LYS	NZ	+1.0
ARG	NE	+1.0

The grid spacing is dynamically determined based on the number of atoms in the input molecule. However, it is possible to specify a fixed spacing in the grid preferences. Be warned, however, that specifying very fine spacings (< 2.0 Angstroms) may consume a surprisingly large amount of memory, (in some cases more than may be available on the computer), which can lead to program failure. This should be considered a very advanced feature to be used only when necessary and when all other data has been saved. Finally, it has been observed in research at OpenEye that adding salt to the electrostatic calculation has a beneficial effect on the grids generated. The salt concentration can be specified in the preferences. The default concentration is 0.04 M. The maximum concentration that can be specified is 0.1 M.

Electrostatic grids (as well as ET grids) are special in that they are always generated with two default contours: a positive one and a negative one. These contours cannot be deleted and new contours cannot be added. The positive contour is colored blue by default and the negative contour is colored red. The contour levels of the two contours are coupled together and so changing one will change the other.

4.11.2 ET

ET (or Electrostatic Tanimoto) grids are generated by OpenEye's BROOD and EON applications. When viewed in VIDA they behave, and are treated essentially the same, as Electrostatic grids.

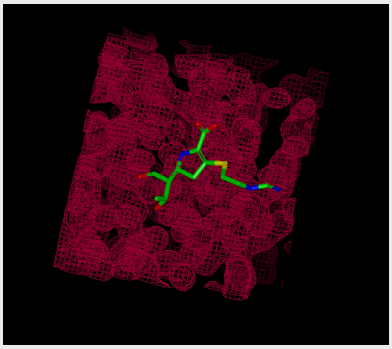
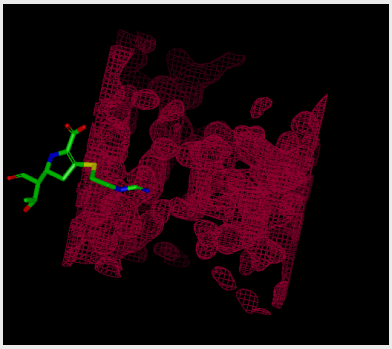
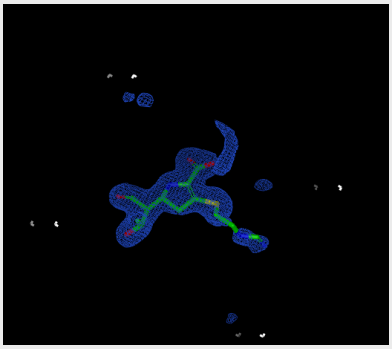
4.11.3 FRED

FRED grids are shape potential grids that are created by OpenEye's FRED application. In VIDA, the behave just like Generic grids except that their default contour color is purple instead of blue.

4.11.4 Generic

Generic grids are effectively the most basic grids and act as the primary grid workhorses. They are completely customizable and have bare-bones settings. They default to having just one contour with a default color of blue. Additional contours can be added as well as removed. These default values can be changed in the *Grids* section of the application preferences.

4.12 Reentrant Grids

		
Infinite grid	Infinite grid rendered at a different center	Non-reentrant grid

Some grids have fixed extents and some grids are infinitely reentrant in space. Reentrant grids obviously cannot be displayed in their entirety, so instead only the region around the center of the current scene is displayed. Grids that are being rendered as reentrant can be distinguished from non-reentrant grids by the absence of grid corners which define the extents of non-reentrant grids.

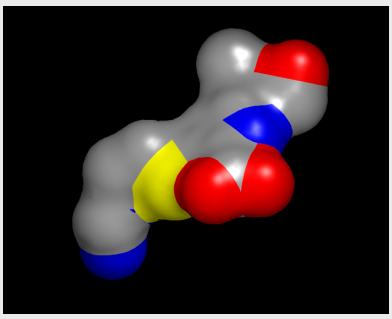
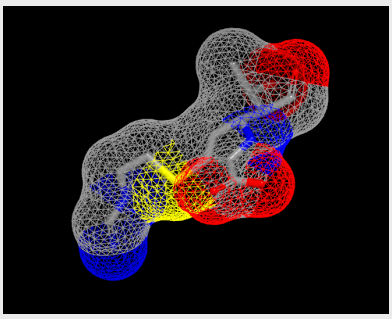
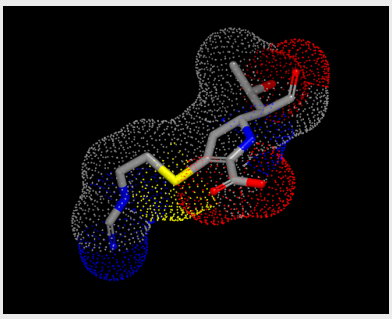
4.13 Surface Visualization

Surfaces are infinitely thin three dimensional connected regions that represent objects such as molecular or accessible surfaces of molecules. Surfaces can be selected (or scribed), cropped, colored, and made transparent in part or in whole.

4.13.1 Display Styles

Surfaces can be visualized in one of three display styles: Solid, Mesh, and Points. These individual display styles can be seen in *Table: Surface Styles*.

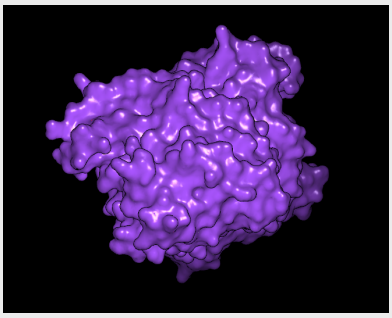
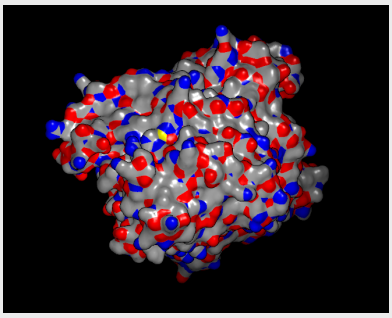
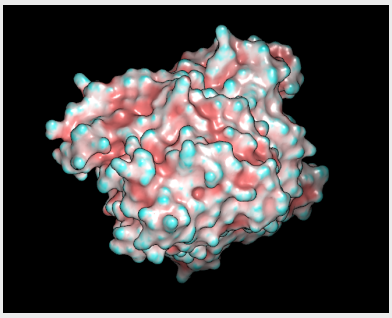
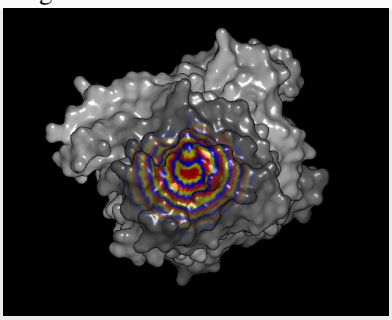
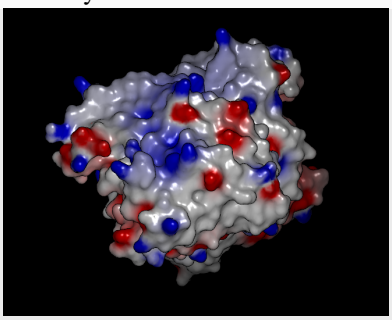
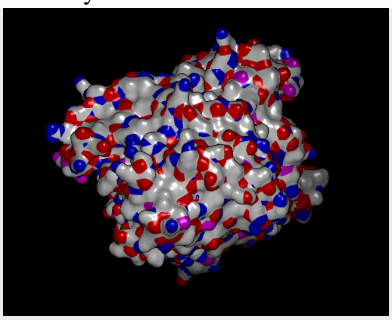
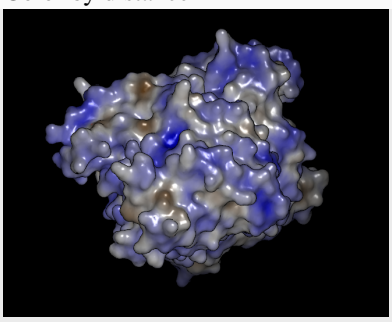
Table 16: Surface Styles

		
Solid	Mesh	Point

4.13.2 Surface Color

There are a number of surface specific coloring schemes in addition to the standard single color scheme and some examples are illustrated in *Table: Surface Colors*. The following schemes are discussed below: atom color, curvature, distance, electrostatics, grid, hydrogen bond potential, hydrophobicity, and surface potential.

Table 17: Surface Colors

		
Single color	Color by atom	Color by curvature
		
Color by distance	Color by electrostatics	Color by hydrogen bond potential
		
Color by hydrophobicity		

Atom Color

The *atom color* scheme colors each vertex on the surface using the color of the nearest atom to that vertex in the molecule that the surface was created from (see *Table: Surface Colors*). This scheme will not work if the surface was not created from a molecule and also if that molecule is not currently present.

Curvature

The *curvature* scheme colors each vertex on the surface according to the curvature of the surface at that vertex using a pink-to-cyan gradient where cyan indicates high convexity and pink indicates high concavity (see *Table: Surface Colors*).

Distance

The *distance* scheme colors each vertex by its distance from either the selected set of atoms or the visible set of atoms (if none are selected at the time of coloring). Each band of color represents a distance of one Angstrom from the selected set. The color banding is continuously repeated as the distance increases, but the colors fade to white with each repetition to show the distance effect (see *Table: Surface Colors*).

Electrostatics

The *electrostatics* scheme colors each vertex on the surface according to the electrostatic potential at that vertex using a red-to-blue gradient from -7.0 to +10.0 (see *Table: Surface Colors*). This range can be changed if desired in the *application preferences* in the *Surfaces* section.

The electrostatic potential at the surface is calculated using OpenEye's **Zap** toolkit (for more information, please see the **Zap** toolkit documentation). By default, VIDA will use the molecule's input partial charges in the calculation (if any were specified). However, if no partial charges were specified, VIDA will assign partial charges using either MMFF94 (the default) or AM1-BCC. The choice of charge model can be specified in the application preferences. If a molecule has greater than 50 heavy atoms, MMFF94 will always be used in place of AM1-BCC regardless of the set preferences.

Proteins have an alternative charging option based on residue information. The charge model can be seen in the table below. This model is used by default (if no partial charges were specified), but it can be disabled in the application preferences.

Residue	Residue Atom	Partial Charge
ASP	OD1, OD2	-0.5
GLU	OE1, OE2	-0.5
LYS	NZ	+1.0
ARG	NE	+1.0

Grid

The *grid* scheme colors each vertex on the surface according to the grid potential at that vertex using a red-to-blue gradient from the minimum grid value to the maximum grid value. The grid potential is determined by mapping the specified grid onto the surface.

Hydrogen Bond Potential

The *hydrogen bond potential* scheme colors each surface vertex according to the hydrogen bond class (Acceptor=Red, Donor=Blue, Both=Magenta) of the nearest atom in the surface's molecule (see *Table: Surface Colors*). This scheme will not work if the surface was not created from a molecule and also if that molecule is not currently present.

Hydrophobicity

The *hydrophobicity* scheme colors each vertex according to a specified hydrophobic color scale based on the OEXLogP values of the underlying atoms. More hydrophobic areas are brown, while more hydrophilic areas are blue. This scheme will not work if the surface was not created from a molecule or if that molecule is not currently present.

Surface Potential

The *surface potential* scheme colors each vertex according to the potential value at that vertex using a red-to-blue gradient from the surface potential minimum to the surface potential maximum. This scheme differs from the *electrostatic* scheme in that the coloring is based on the stored potential values as opposed to calculated potential values. Typically, surfaces generated within VIDA will not have any stored potential values. However, surfaces created externally using OpenEye's **Spicoli** toolkit, for example, can store values in the potential field and as such can be colored accordingly.

4.13.3 Selection

For surfaces created from molecules, the minimum selectable part is the surface corresponding to a single atom. A line of atom surfaces can be selected by selecting one spot on a surface and then by holding down the *Shift* key when selecting another. This starting selection can be grown (and shrunk) using the *Selection* pane in the Style Control. Double-clicking on the surface will select the entire surface. For surfaces created without a molecule (usually outside of VIDA), the minimum selectable part of a surface is a single vertex, and the above operations work on vertices instead of atoms.

4.14 Symmetry

VIDA supports symmetry operations. This support includes a submenu under the *Edit* menu which controls displaying symmetry in the 3D window, and some items in the popup menu of the list window which allow setting or editing symmetry parameters.

Symmetry information may be read from certain files. When VIDA reads PDB-format files which have valid CRYST1 records, symmetry information is taken from that record for all molecules read from that file. Similarly, when VIDA reads MTZ-format reflection files the symmetry information is read from that file and, of course, VIDA will read symmetry information if it is present in an `.oeb` file.

When symmetry display is enabled, all symmetry operators which would result in a molecule within a certain distance (the *symmetry radius*) of the screen center are displayed. This means that as the view is translated, certain symmetry *replicates* will appear and disappear as they move into or out of the symmetry zone. There is, therefore, no need to manually re-calculate the symmetry after translating the model.

4.14.1 Symmetry Menu

The Symmetry menu is located in the *Style* menubar. The items in the symmetry menu are:

- *Edit cell* This item allows editing or setting of the unit cell and space group parameters of the active object, which will generally be either a reflection data set or a molecule. In addition to the interactive symmetry display, information entered here will also be used for the CRYST1 record if the molecule is written as a PDB file.
- *Show unit cell* This toggles the display of the crystallographic unit cell of molecules which have symmetry information specified.
- *Show symmetry* This toggles the display of symmetry-related molecules, hereafter called "replicates". If enabled, *all* visible molecules with symmetry information will display symmetry.
- *Realize symmetry* The symmetry display above is only for display purposes. If the user wishes to actually expand a molecule to produce one with "real coordinates" for the replicates, the symmetry must be "realized". This option will prompt the user to determine if the current molecule(s) should be symmetry-expanded "in place" or if a new copy of the molecule(s) should be created and then symmetry expanded. Only the symmetry operators currently enabled are used in the expansion.
- *Symmetry radius* All symmetry operators which would produce a molecule within the symmetry radius of the screen center are shown. With a large symmetry radius, this may result in more replicates than symmetry operators, as each operator may be used more than once with different translations. As a special case, if the symmetry radius is 0.0, then each operator is used exactly once, with a translation which places the center of mass of the molecule in the default unit cell (i.e. the cell at the origin).
- *Symmetry operators* This submenu lists the symmetry operators for the *focused* molecule. It is possible to toggle individual symmetry operators on and off. This option is very useful for (e.g.) producing the biologically active dimer using crystal symmetry. If there are no operators listed in this submenu, then either a) no object has the focus, b) the focused object is not a molecule, or c) the focused molecule does not have symmetry information. Currently, symmetry information is only read from the CRYST1 record of PDB files.
- *No symmetry color* No special coloring will be used for symmetry replicates.
- *Single symmetry color* A single (distinct) color will be used for all symmetry replicates.
- *Color by operator* Each symmetry operator is assigned a color. All symmetry replicates resulting from the same operator (e.g. X, -Y, Z) have the same color no matter which unit cell they are in
- *Unique color* Each replicate is given a distinct color, so that no two replicates have the same color. Since the number of replicates can change depending on the symmetry radius and/or the current center, this has the somewhat unfortunate side-effect that occasionally translations cause every molecule to be recolored.

Note

After realizing symmetry, if "Show symmetry" is left on there could be a very large number of molecules on screen as each new molecule is also symmetry expanded.

CHAPTER

FIVE

2D DISPLAY

2D molecular depictions are provided using OpenEye's **OEDepict** toolkit. A large number of options are available in the applications preferences to customize how the depictions are drawn.

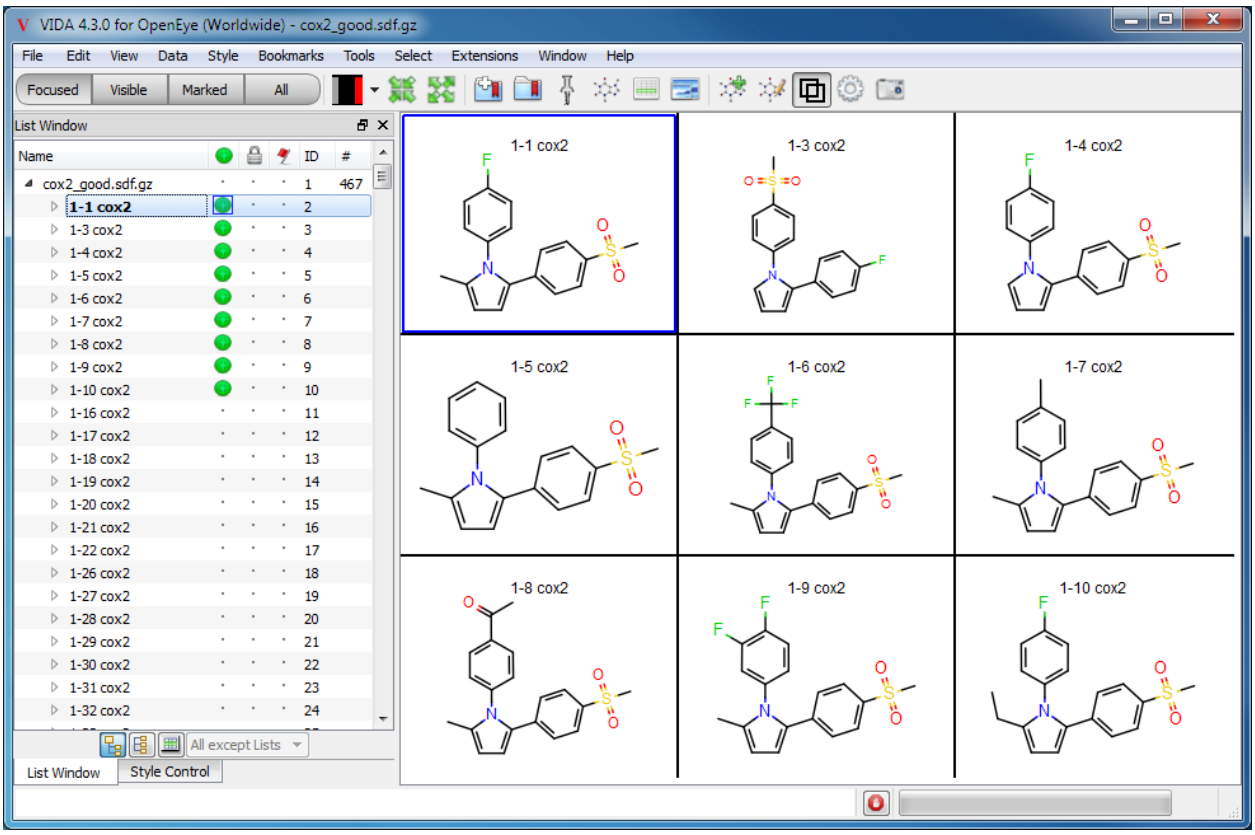
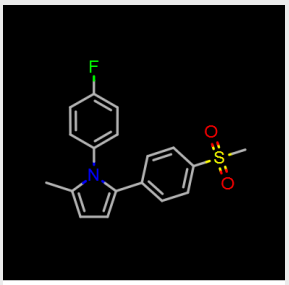
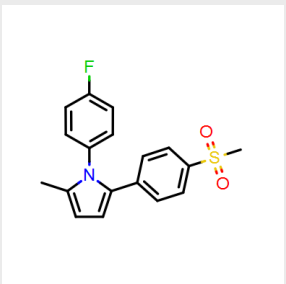
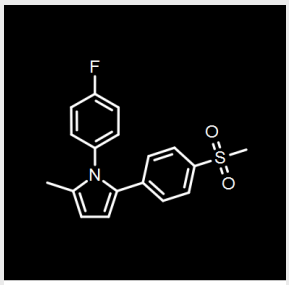
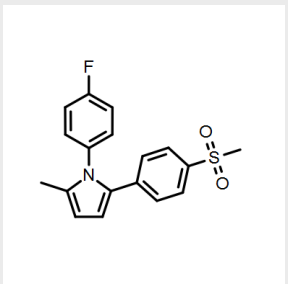


Fig. 1: **Molecule Depiction**

Depictions are drawn in color on a white background by default as seen in *Figure: Molecule Depiction*, but other color schemes are available and can be seen in *Table: Depiction Colors*.

Table 1: **Depiction Colors**

			
Color on black	*Color on white*	*White on black*	*Black on white*

As might be expected, 2D depictions of this nature only make sense for small molecules (< 255 atoms in this case). For proteins, an amino acid sequence view is displayed instead (see *Figure: Protein Sequence*).

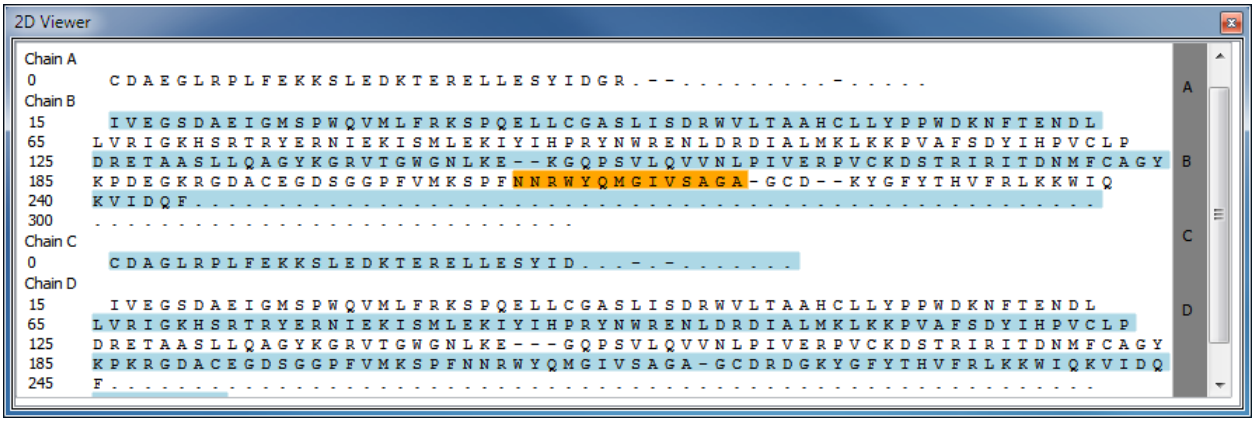


Fig. 2: **Protein Sequence**

5.1 Preview vs. Viewer

The 2D display comes in two flavors - the 2D Preview and the 2D Viewer. There are two differences between these two windows. First, the 2D Preview can only show a single molecule – the *Focused* one – whereas the 2D Viewer can show several in a grid if the application is in Tiled mode. Secondly, the 2D Viewer can be set as the main window, but the 2D Preview cannot. The 2D Viewer is shown in *Figure: Molecule Depiction*.

The 2D Preview is best suited for use as a small secondary display of the *Focused* molecule while otherwise browsing. The 2D Viewer often requires more screen real estate and is usually best suited as the main window.

5.2 Interaction

2D selection behaves much like selection in the 3D display. Left clicking will select an individual atom, bond, or residue (if in sequence mode). When viewing a depiction, the right mouse button allows lasso style selection, as in the 3D display. When viewing a sequence, holding down the *Shift* key when selecting will allow selection of a continuous range of residues.

The current selection is reflected in the coloring of the 2D depiction. There may be slight variations between the depicted selection in 2D and 3D due to the difference in how hydrogens are drawn.

5.3 Advanced Depiction Options

Some advanced options for molecular depictions are available in the Advanced Depiction Options dialog (see *Figure: Advanced Depiction Options*), which can be accessed through the *Data* menu.

Through this dialog, it is possible to add property maps to depicted molecules in the 2D Viewer and Spreadsheet using the **Grapheme** toolkit. Additionally, this dialog allows the designation of reference template molecules that can be used to align 2D depictions to a common orientation. The reference templates can also be used to highlight matching substructures in depicted molecules.

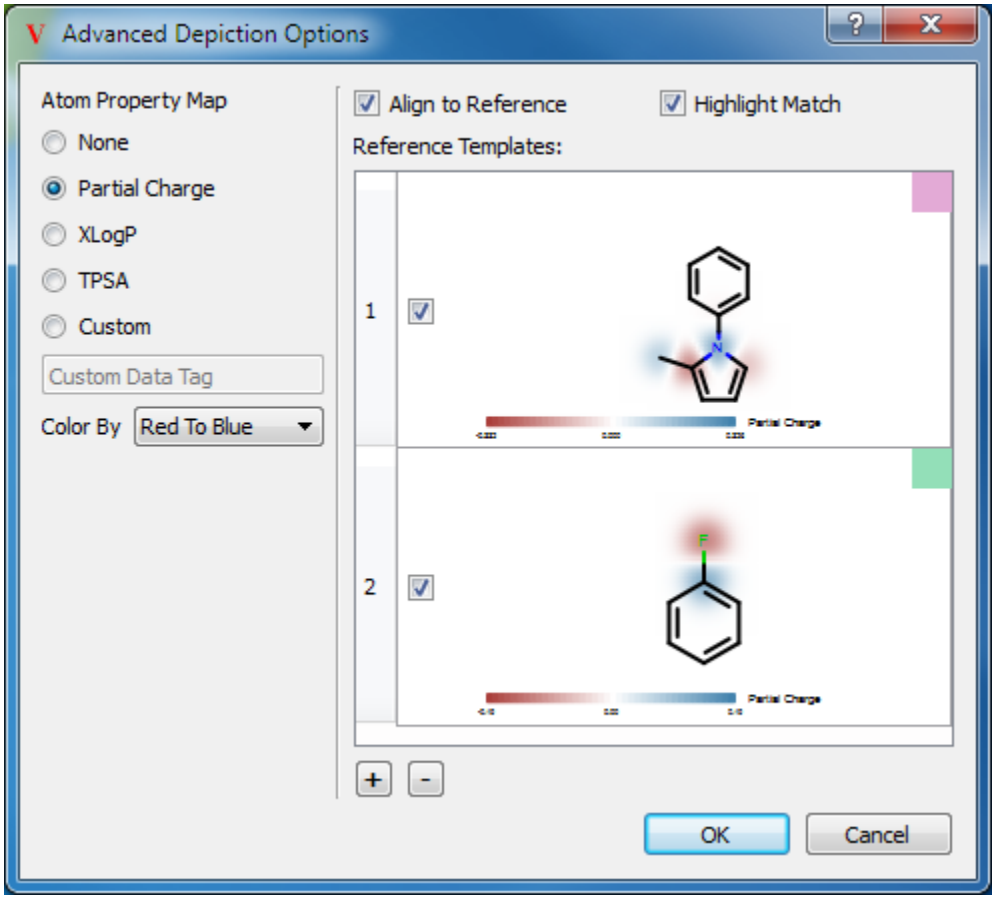


Fig. 3: **Advanced Depiction Options**

Atom property map displays are enabled by choosing one of the Atom Property Map options on the left of the dialog. Three of the properties, *Partial Charge*, *XLogP*, and *TPSA*, VIDA will calculate internally. To display property maps for other atom-level properties, select the *Custom* option and specify a generic data tag to use for property map displays. The atoms must have real-valued generic data on them for the *Custom* option to work.

Reference molecule templates can be pasted into the dialog or sketched, and multiple templates can be specified. The first template matching a molecule will be used for alignment and highlighting in that molecule's depictions.

If *Align to Reference* is checked, molecules in the 2D Viewer and spreadsheet will be aligned to match the orientation of the first matching reference template. If *Hightlight Match* is checked, the part of the depicted molecule that matches the template will be highlighted using the color shown in the Reference Templates list.

Individual templates can be enabled or disabled using the checkbox next to the molecule in the Reference Templates list. New templates can be added, and selected templates deleted, using the "+" and "-" buttons below the Reference

Templates list.

CHAPTER

SIX

SPREADSHEET

The spreadsheet window is used to view, compute, and analyze data associated with molecules and can be seen in *Figure: Spreadsheet Window*. The spreadsheet window contains four initial spreadsheets by default: *Molecules*, *Proteins*, *Atoms* and *Residues*. The *Residues* and *Atoms* spreadsheets behave slightly different than the other two and are described later. Additional spreadsheets can be added during use such as the *IC50* spreadsheet shown in *Figure: Spreadsheet Window*.

The *Molecules* spreadsheet is populated with all currently loaded small molecules (fewer than 255 atoms). Conversely, the *Proteins* spreadsheet is populated with all of the currently loaded molecules that have more than 255 atoms.

The columns and their associated values are populated from SD data. SD data can be found in both SDF and OEB files. Additional spreadsheet columns can be added from generic data, by computing Python expressions or by importing external data from a comma-separated value (.csv) or tab-separated value file (.txt).

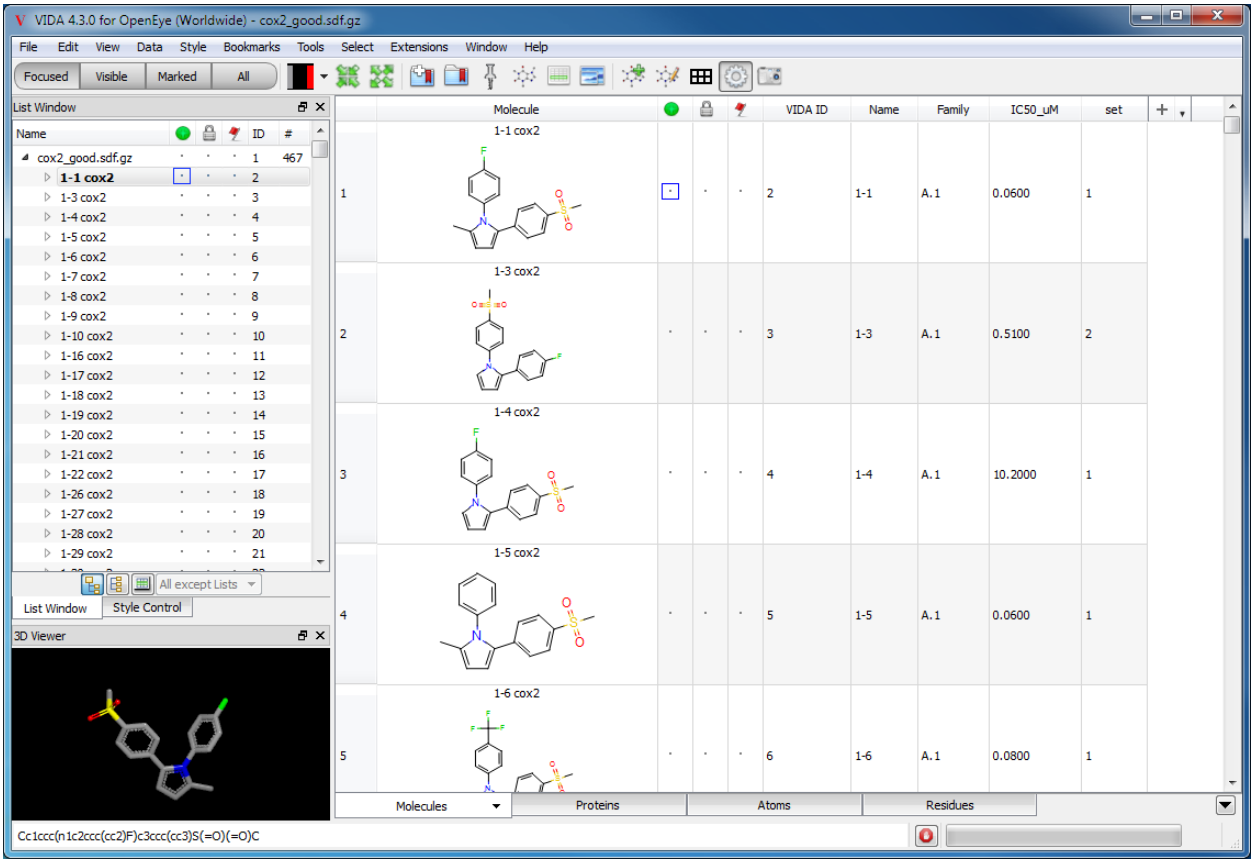


Fig. 1: **Spreadsheet Window**
The spreadsheet being used as the main window of the application

6.1 Basic Usage

Columns are divided into two basic types: data and property. Data columns are those found in SD data and/or imported files. Property columns include:

- Depiction - This column contains either a 2D depiction of the associated molecule (for small molecules) or an interactive 3D rendering of the protein backbone (for proteins).
- VIDA ID - This column contains the unique identifier assigned to that object by VIDA.
- VIDA Name - This column contains the title of the molecule as displayed in VIDA.
- Visible - This column contains an indicator (green dot) showing whether the molecule is *Visible*.
- Marked - This column contains an indicator (red flag) showing whether the molecule is *Marked*.
- Locked - This column contains an indicator (padlock) showing whether the molecule is *Locked*.

Clicking on any data cell in the spreadsheet will cause the associated molecule to become the *Focused* molecule. Double-clicking on a cell in the spreadsheet enables modification of the data value in that particular cell. Some columns cannot be changed, such as the *VIDA ID* or *Depiction* columns.

Clicking the mouse on a molecule's depiction and dragging the mouse to another application will transfer that image to that application. This provides an easy way to transfer molecular images to other documents.

Visible, *Marked*, and *Locked* properties can be toggled for selected rows by clicking in their columns, or by pressing the keys, V(isible), M(ark), or L(ock), respectively. When a range of rows is selected, toggling properties in any of the

selected rows will toggle the same property in all of the selected rows.

6.2 Sorting

A spreadsheet can be sorted by clicking on a column header which will sort the current spreadsheet according to the data values in the associated column. If the data in the column to be sorted contains both numeric and non-numeric values, each set (numeric and non-numeric) will be sorted independently and then concatenated together. Clicking again reverses the sort order.

Multiple columns can be sorted simultaneously by selecting the "Sort Spreadsheet…" option from the top level *Data* menu. This will launch a dialog that can be seen in *Figure: Sorting*. This dialog can also be launched by right-clicking on any column header and selecting the "Sort Spreadsheet…" option from the generated popup menu.

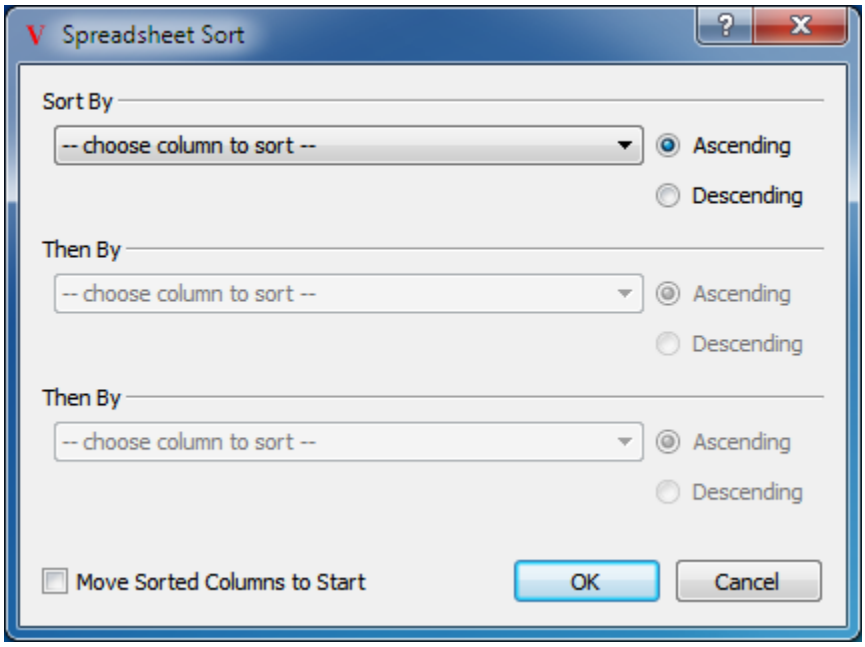


Fig. 2: **Sorting**
Simple dialog to control how the current spreadsheet is sorted

This sorting dialog allows for the selection of up to three columns. The spreadsheet will initially be sorted by the first column selected, with ties being broken by the remaining columns. The sorted columns can be moved to the front of the spreadsheet by selecting the "Move Sorted Columns to Start" option in the dialog.

The first time the spreadsheet is sorted, it may take longer than expected as the data values need to be retrieved from the molecule and any expressions must be calculated. The progress of the sort is indicated by the progress bar at the bottom of the application. If desired, the sort can be aborted by clicking on the stop button to the left of the progress bar.

6.3 Displaying Data

The visibility of columns can be toggled by selecting the associated column name from the right-click popup menu generated by clicking on any column header. Data column names can be found in the "Column Visibility" submenu. Hiding the depiction column can allow for more rows to be displayed at the same time.

Certain molecular properties, such as molecular weight, can be added as new columns by selecting the desired option from the "Molecular Properties" submenu in the right-click popup menu. Some of these properties, such as Energy,

might not be set on the molecule and as such the associated cell will contain an empty value.

6.4 Organizing Columns

Columns can be organized in the spreadsheet by selecting the "Organize Columns" option in the top level *Data* menu. This will launch a dialog which can be seen in *Figure: Column Controller*.

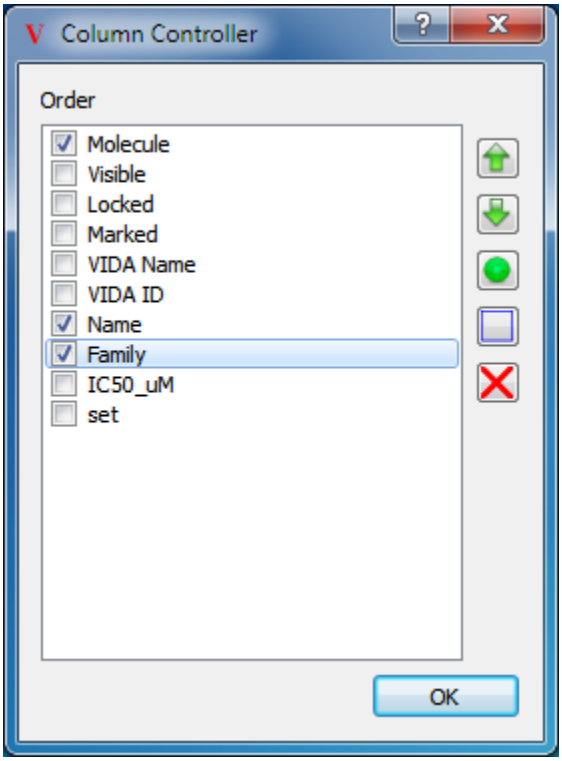
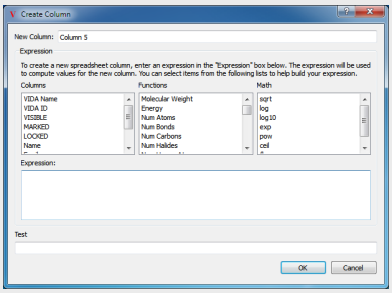


Fig. 3: **Column Controller**
The display of individual columns can be customized using this dialog

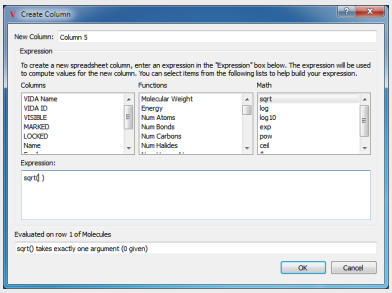
The order of the columns can be controlled by selecting individual columns and then clicking on the up or down arrow buttons to change that columns position in the order. Furthermore, columns can be deleted by clicking on the button with the red X icon. Please be aware that deleting a column will delete the associated data from all currently loaded molecules and spreadsheets.

6.5 Creating New Columns

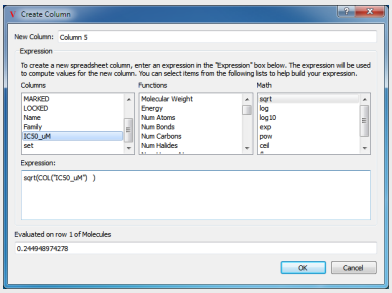
Columns for common molecule and atom properties can be easily added to a spreadsheet using the pull-down menu in the far right column of the spreadsheet header. The Molecule Properties (or Atom Properties for the Atoms spreadsheet) submenu of this menu offers large selection of common properties that can be used to make new spreadsheet columns.



Creating a new column



Using a math function in the expression



Referencing another column

For more complicated properties, new columns can be added to the spreadsheet with values that are based on, or calculated from, molecular properties or existing spreadsheet data. To create a new columns, select the "Create Column" option in the top level *Data* menu, or click the "+" button in the spreadsheet header. This will launch a dialog to guide the process and can be seen in *Figure: Create Column*.

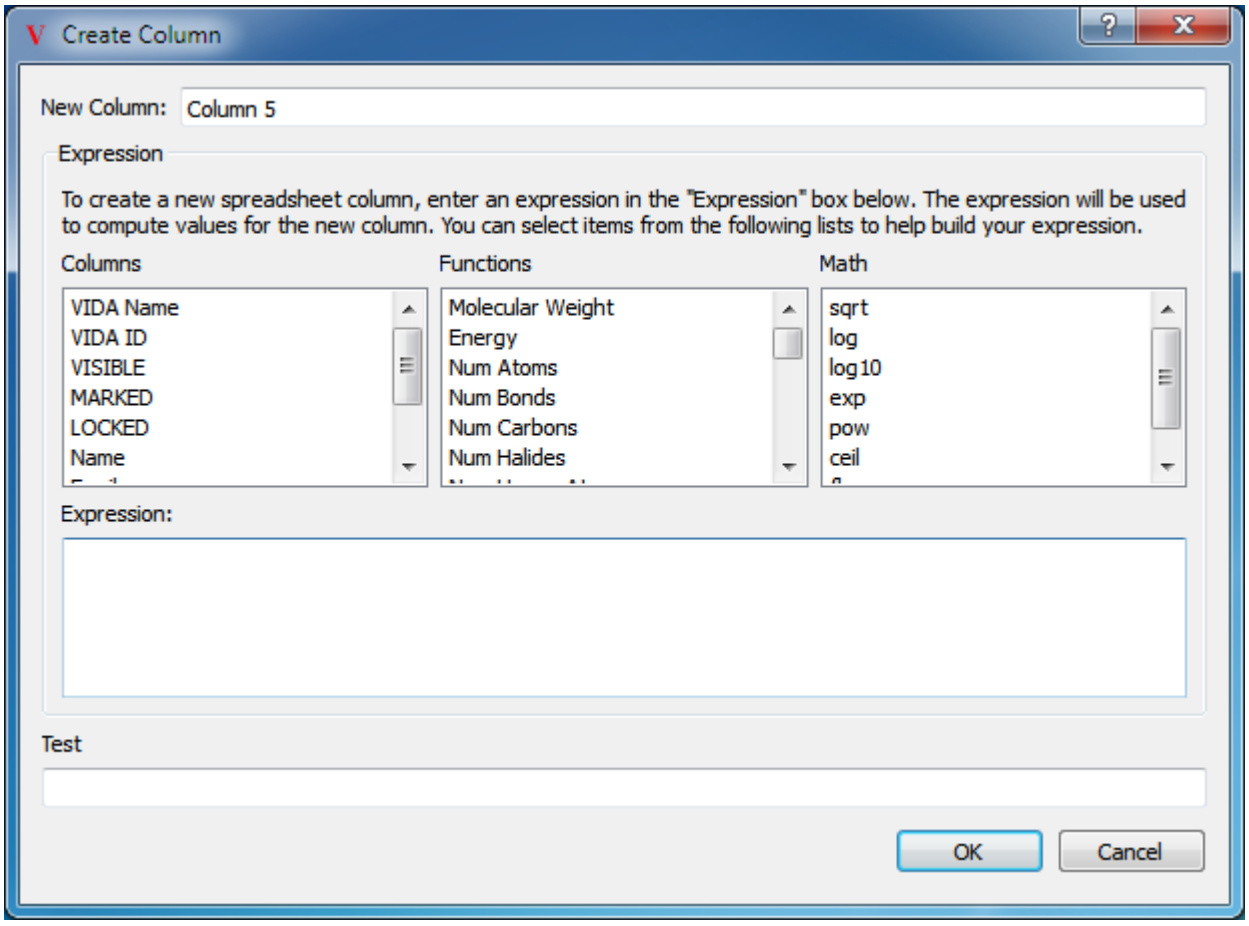


Fig. 4: **Create Column**
Create a new spreadsheet column by specifying a Python expression

The text entry field at the top contains the name for the new column. A new, empty and editable column can easily be created by supplying a new column name and then clicking the "Create" button.

Beneath the name is a collection of three lists containing functions which can be used in building the expression to be evaluated when populating the column. The first box contains a list of the already existing columns which can be used as part of an expressions. The second box contains a list of available molecular properties. The third box contains a list of available mathematical functions.

Beneath the boxes is a large text entry window which will contain the desired Python expression used to calculate the new data values. Any valid Python expression can be specified here. Just beneath this window is a test field which displays the output of running the specified expression on the first row in the current spreadsheet.

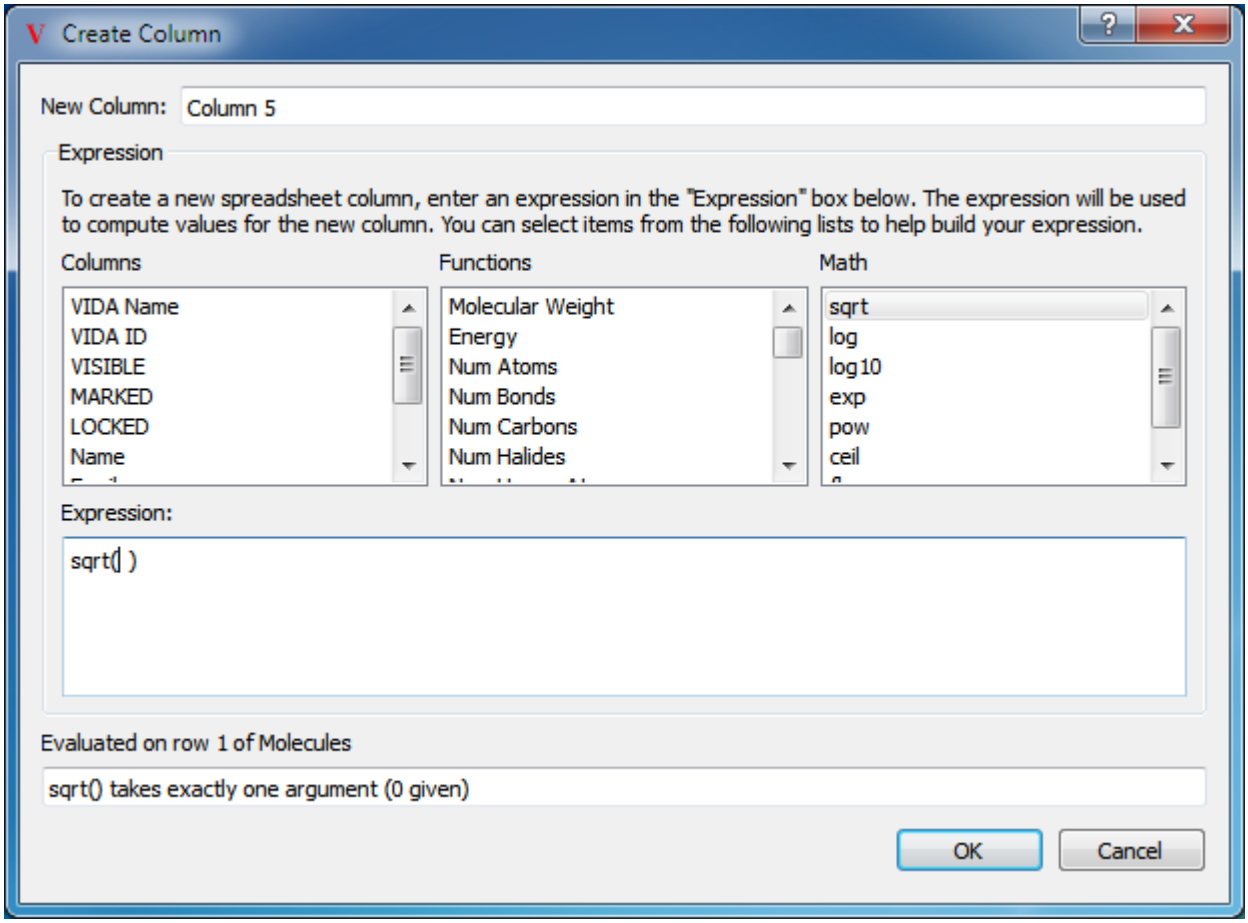


Fig. 5: **Math Functions**
Specifying a math function to be used in the expression

Any of the previously mentioned functions can be added to the Python expression simply by selecting the desired function in the appropriate drop down box. If that function requires additional input (such as functions like *sqrt*), the cursor in the expression editor window will automatically be placed inside the function. Functions can be combined with standard math operators such as "*", "+", "-", and "/". If a function requires additional input that is not provided, an error will be displayed in the Test section at the bottom of the dialog as seen in *Figure: Math Functions*. Frequently, the desired input is contained within another column which can be referenced by selecting the desired column from the third drop down box or simply by typing "COL('COLUMN NAME')" as seen in *Figure: Referencing a Column*.

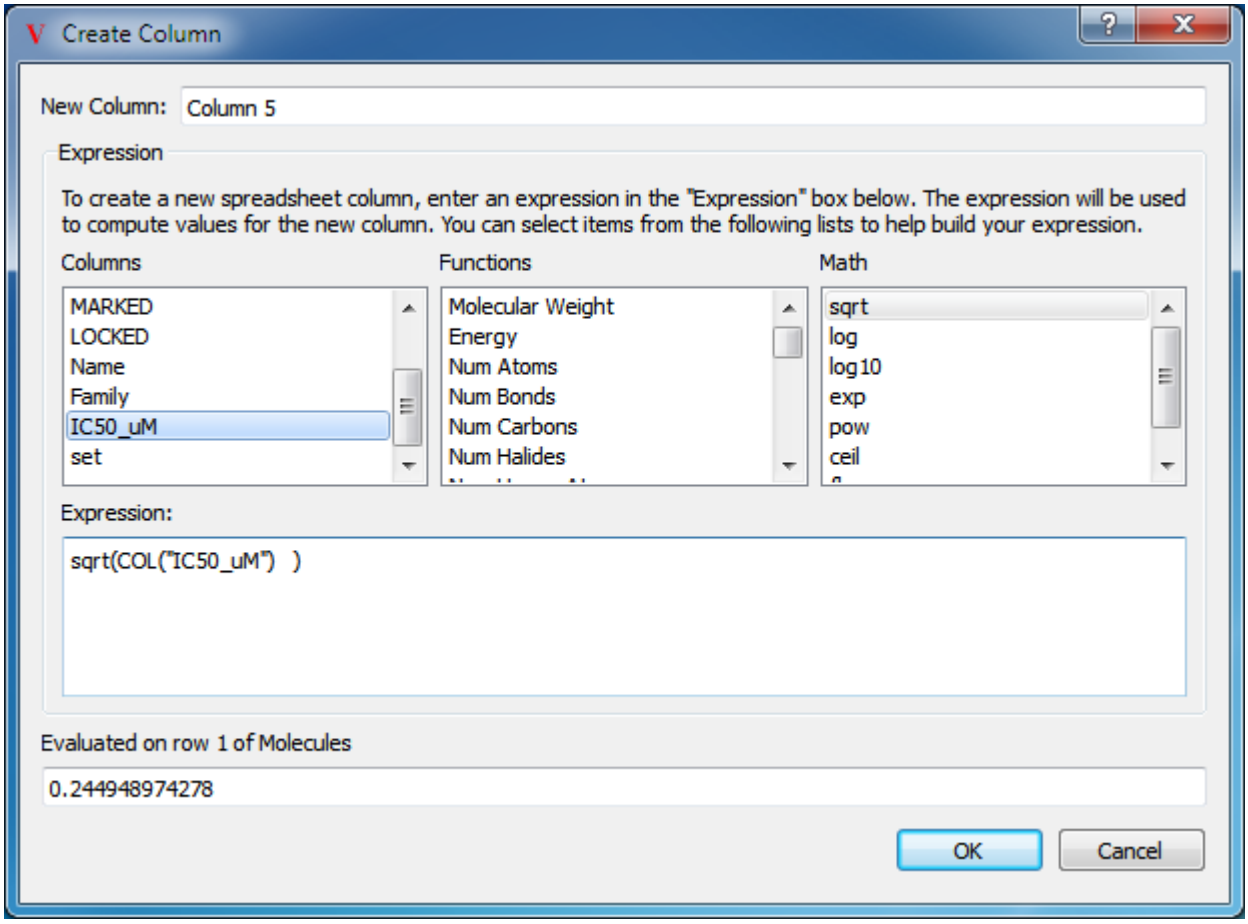


Fig. 6: **Referencing a Column**
Referencing another column to provide data for the expression

Once the desired expression has been assembled, the column can be created and added to the spreadsheet by clicking on the "OK" button.

6.5.1 Math Functions

A number of math functions are available for use in spreadsheet expressions as described in the previous section. The currently available math functions can be found in *Table: Math Functions*.

Table 1: Math Functions

Name	Description
sqrt(x)	Return $\sqrt{x}$.
log(x[,base])	The logarithm of x to the given base. If the base is not specified, returns ln(x).
log10(x)	The base 10 logarithm of x.
exp(x)	Return e^x.
pow(x,y)	Return x^y.
ceil(x)	Return the ceiling of x as a float. This is the smallest integral value >= x.
floor(x)	Return the floor of x as a float. This is the largest integral value <= x.

6.5.2 OEChem Functions

A large number of molecular properties are available for use in spreadsheet expressions as described above. The currently available molecular properties can be found in *Table: OEChem Functions*.

Table 2: OEChem Functions

Name	Description
Molecular Weight	Molecular weight.
Energy	Molecular energy of the molecule as specified in the input file.
Num Atoms	Number of atoms in the molecule.
Num Bonds	Number of bonds in the molecule.
Num Halides	Number of halogen atoms in the molecule.
Num Carbons	Number of carbon atoms in the molecule.
Num Heavy Atoms	Number of heavy atoms (non hydrogen) in the molecule.
Num Hetero Atoms	Number of hetero atoms in the molecule.
Num Hydrogens	Number of hydrogen atoms in the molecule.
Num Hydrogen Acceptors	Number of hydrogen bond acceptors in the molecule.
Num Hydrogen Donors	Number of hydrogen bond donors in the molecule.
Carbon-Hetero Ratio	Ratio of carbons to hetero atoms in the molecule. Returns 1 if there are no carbons in the molecule.
Num Rigid Bonds	Number of rigid bonds in the molecule.
Num Rotatable Bonds	Number of rotatable bonds in the molecule.
Formal Charge	Sum of the formal charges on all atoms.
Num Formal Charges	Number of atoms with a specified formal charge.
Partial Charge	Sum of the partial charges on all atoms.
Is Chiral	1 or 0, indicating whether the molecule is chiral.
SMILES	SMILES representation of the molecule.
Molecular Formula	Molecular formula for the molecule.

6.6 Filtering

New spreadsheets may be generated from an existing one by means of filtering. The mechanism to filter a spreadsheet is very similar to the creation of a new column. To create a filtered spreadsheet, select the "Filter Spreadsheet…" option from the top level *Data* menu, or right-click in the header of the spreadsheet and choose "Filter…". This will launch a dialog which can be seen in *Figure: Filter Dialog*. This dialog will assist in the creation of a new filtered spreadsheet based on the currently viewed spreadsheet. Please note, that it is possible to create filtered spreadsheets on other filtered spreadsheets. The result of a filtering operation can be seen in *Figure: Filtered Spreadsheet*.

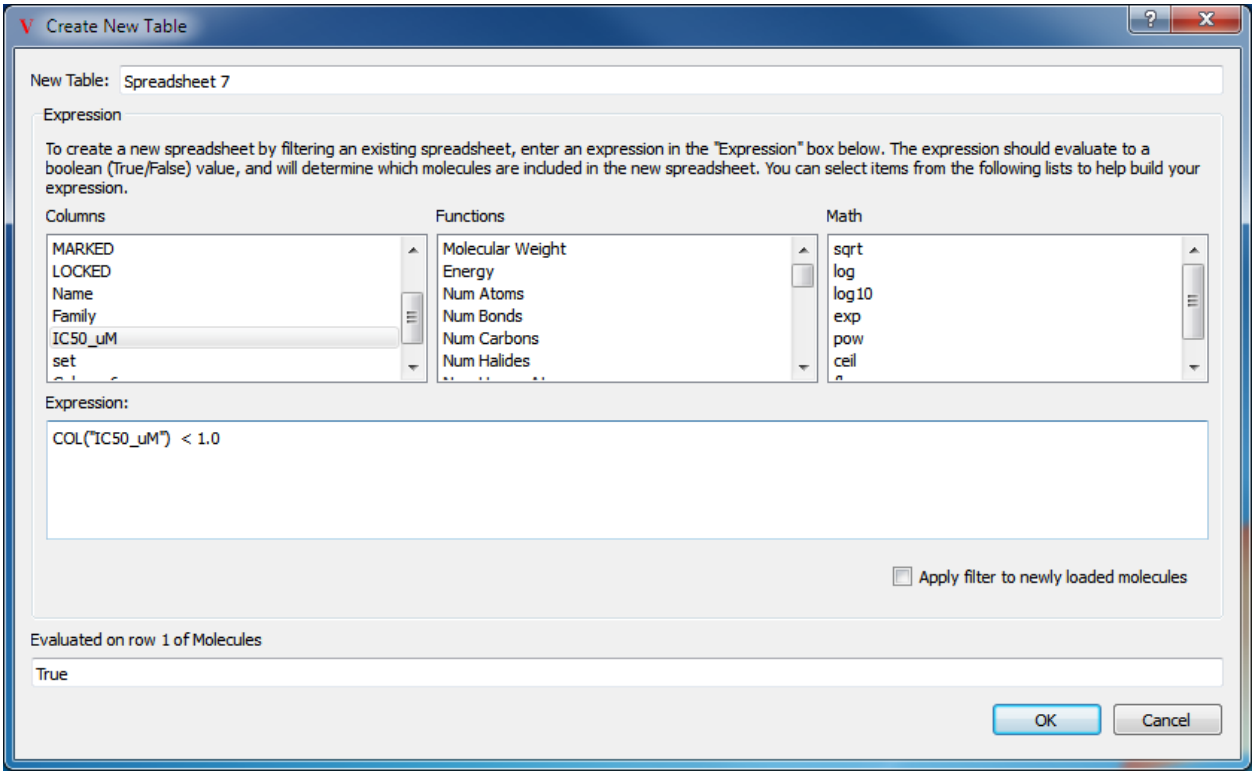


Fig. 7: **Filter Dialog**
Create a new spreadsheet by filtering on a specified column

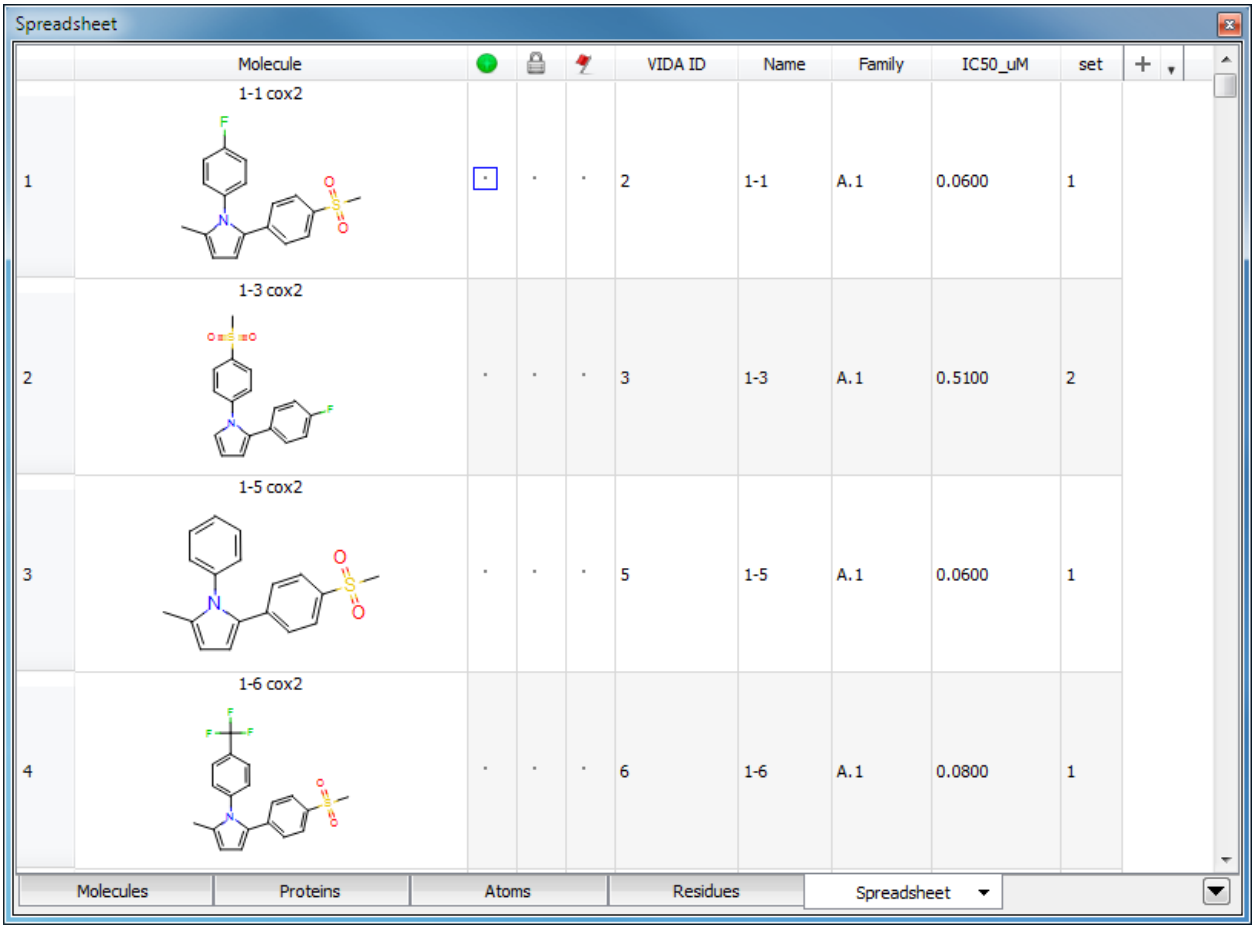


Fig. 8: **Filtered Spreadsheet**
A new spreadsheet created by filtering on the values in a specified column

Filter expressions are very similar to column generation expressions, except that the return value of a filter expression determines whether or not that row will be included in the new spreadsheet. Rows for which the expression returns either *False* or *0* will not be included.

A wide variety of filters are possible through the judicious use of Python in the expression. For instance, the following expression will filter only those matching the specified text string:

```
"<100" in str(COL("IC50_uM"))
```

6.7 Statistics

Statistics for each numeric column can be viewed in the spreadsheet window. The display of statistics is controlled by toggling the "Show Statistics" option in the top level *Data* menu. Toggling this option controls the display of statistics for just the currently viewed spreadsheet.

The following statistical information is computed for each numeric column:

- *min* - The minimum value of the column.
- *mean* - The maximum value of the column.
- *stddev* - The standard deviation of the column.

- *skewness* - The skewness of the column.
- *kurtosis* - The kurtosis of the column.

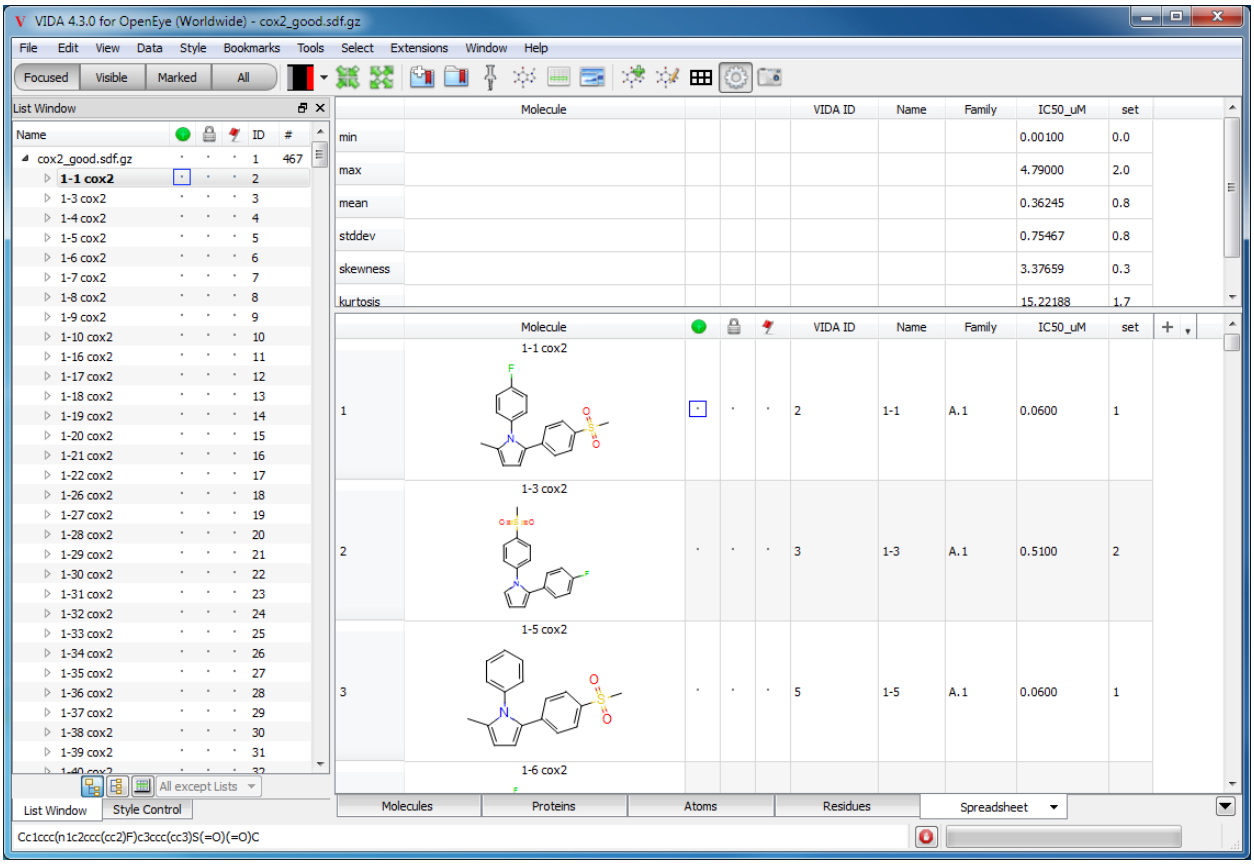


Fig. 9: **Statistics**
Displaying statistical information on the associated spreadsheet

The statistics display is a dynamic view of the spreadsheet data and therefore when data is altered in the spreadsheet the statistics view is automatically updated. For large data sets, this may take some time and could cause editing the spreadsheet to be cumbersome. To avoid this problem, it is best to hide the statistics display until the editing process is complete.

6.8 Formatting Columns

There are a variety of different display styles for spreadsheet columns which can be edited on an individual basis. The customizable styles include color, font, significant digits, and cell height. The style for individual columns can be edited by selecting the "Format Columns…" option in the top level *Data* menu. This will launch the dialog seen in *Figure: Column Formatting*.

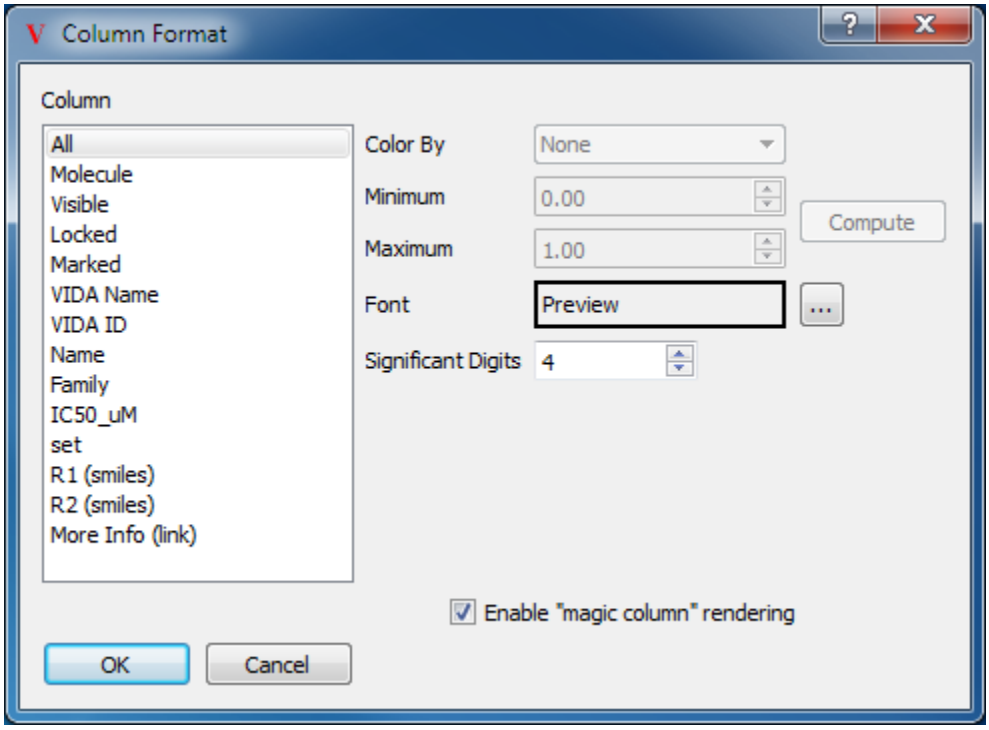


Fig. 10: **Column Formatting**
Many formatting options are available for each individual column

On the left hand side of the dialog is a list of all the currently available spreadsheet columns, including an *All* option. Any formatting changes made in this dialog will be applied to the selected columns in this list.

On the right hand side of the dialog is a collection of formatting and display options. At the top is a drop down menu of available coloring schemes which include:

- Red to Blue
- Blue to Red
- Rainbow
- Rainbow Backwards
- Red Yellow Blue
- Grey to Green

The selected coloring scheme will be applied to the column using the scale specified beneath the drop down menu determined by the minimum and maximum values. The default values are 0.0 and 1.0 respectively, but the current minimum and maximum values for the selected column can be computed by clicking on the "Compute" button to the right of the minimum and maximum fields.

The "Enable 'magic column' rendering" option allows specialized rendering of data, based on column names and cell contents. With this option enabled, data in any column with a name containing "(smiles)" will be interpreted as SMILES strings, and displayed as chemical structures. Data in any columns with "(link)" in the name will be interpreted as URL's. The name of the column, minus the "(link)" string, will be displayed in each cell as an active hyperlink. URL's in columns without "(link)" in the column name will be displayed as raw hyperlinks. See *Figure: "Magic Column Rendering"*.

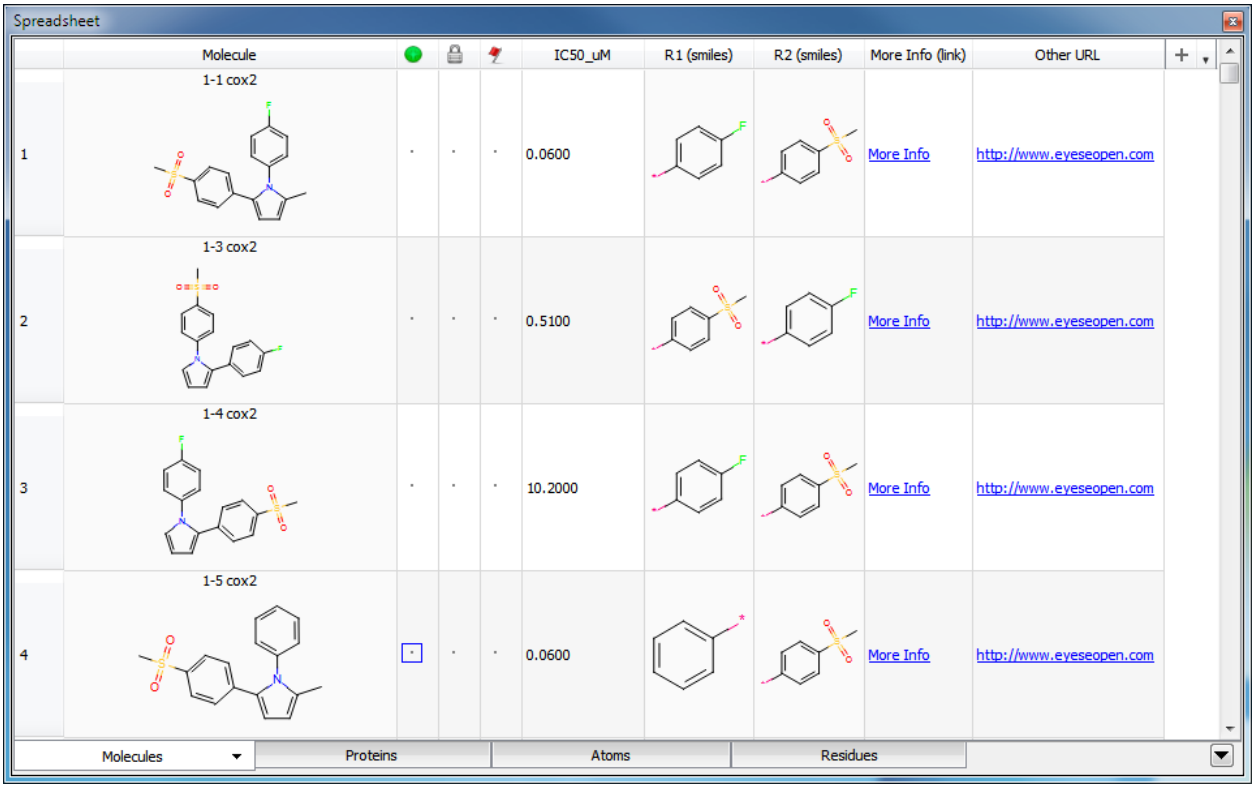


Fig. 11: **Magic Column Rendering**
With "magic column" rendering enabled, columns containing "(smiles)" or "(link)" in the column name will be displayed as chemical structures or hyperlinks, respectively.

Common HTML tags can be rendered in spreadsheet cells, with the notable exception of image (<img>) tags, which will not work.

6.9 Advanced Depiction Options

The advanced depiction options described in the *2D Display* chapter apply to structure depictions in the spreadsheet, as well as the 2D Display. See *Figure: "Template Highlighting"* and *Figure: "Atom Property Maps"* for examples.

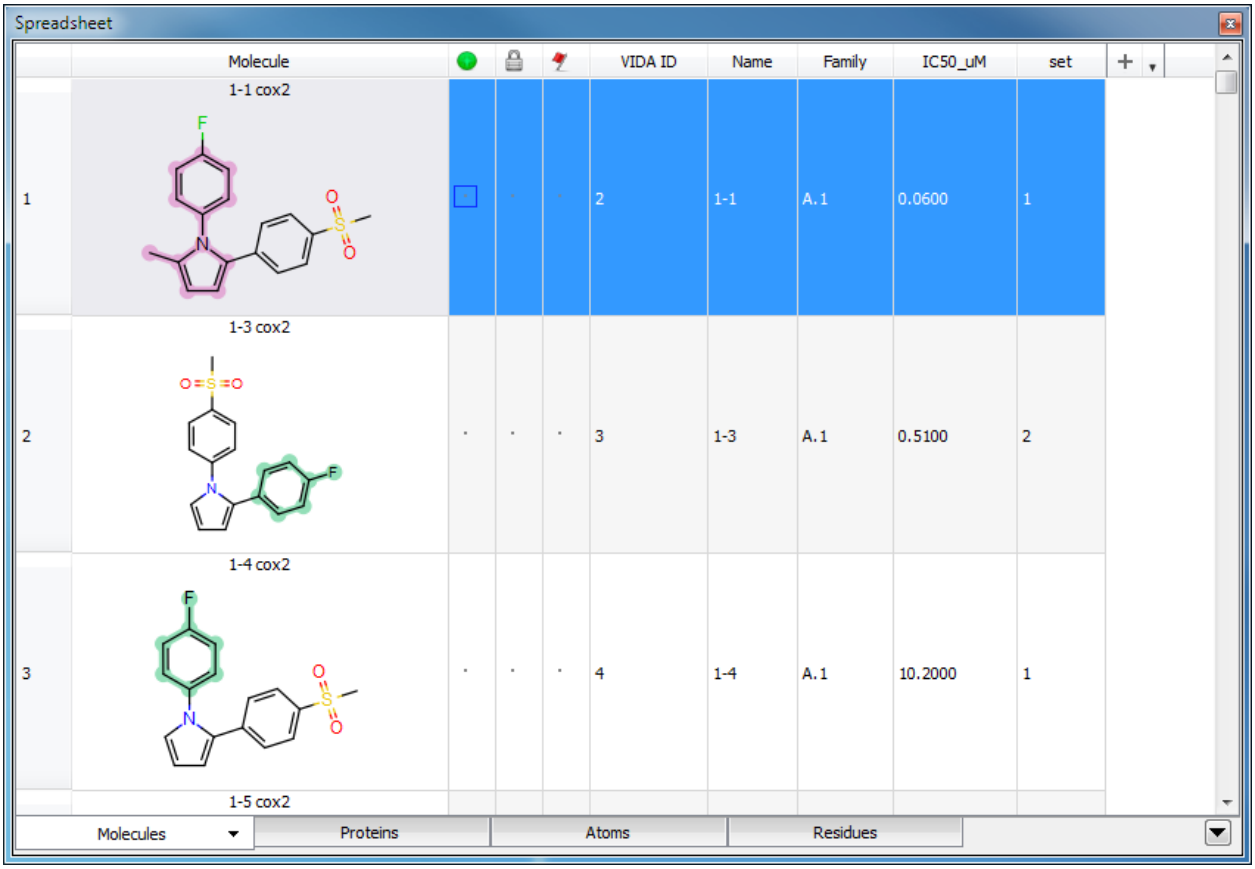


Fig. 12: **Template Highlighting**
Spreadsheet depictions with template highlighting enabled

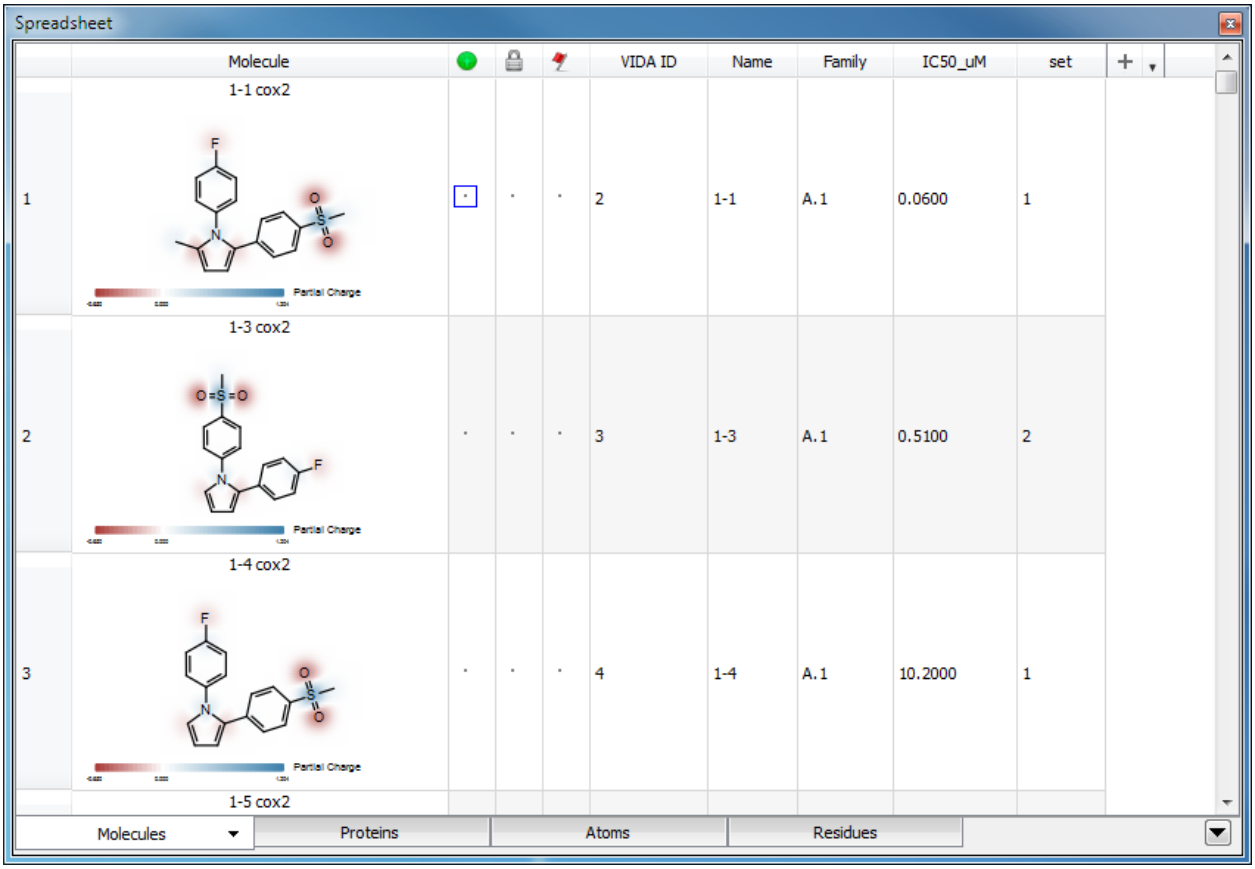


Fig. 13: **Atom Property Maps**
Spreadsheet depictions with Partial Charge property maps enabled

6.10 Importing/Exporting

6.10.1 Importing

Spreadsheet data can be imported from external files in a variety of formats including comma-separated values (.csv) and tab-delimited values (.txt). To import a file, select the "Spreadsheet…" option in the *Import* submenu of the top level *File* menu. This will launch a dialog to assist in the import process which can be seen in *Figure: Importing (1)*.

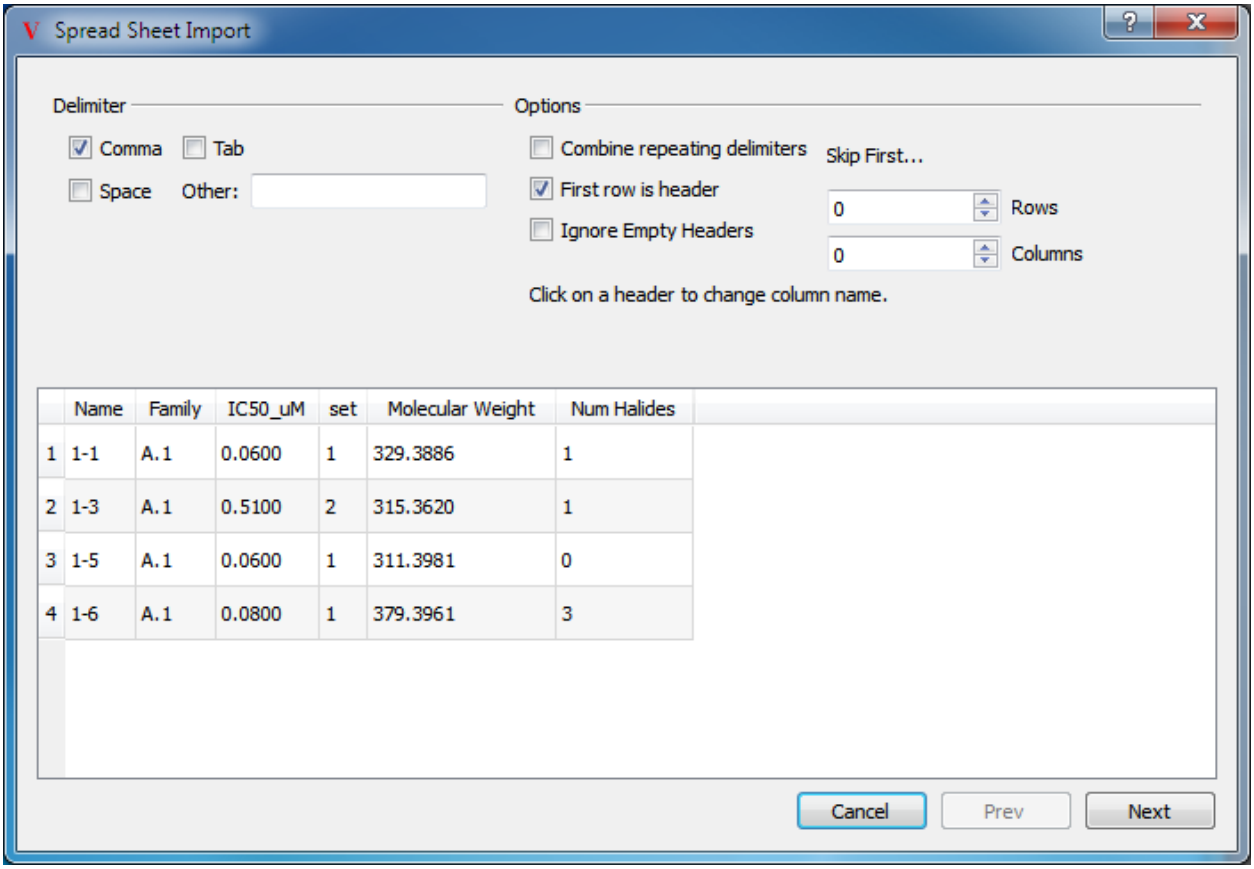


Fig. 14: **Importing (1)**
Specifying the delimiter type and other options

At the top of the dialog are a number of options to specify the format of the file to be imported and at the bottom is a quick preview of what VIDA thinks the first five rows are. Note that in the figure only four rows are displayed in the preview because the first row is being interpreted as the header. The preview will update dynamically to match any changes made to the options at the top.

Once the preview of file to be imported is satisfactory, click on the *Next* button to specify how the rows are to be matched to currently loaded molecules and what to do with rows that do not match (see *Figure: Importing (2)*).

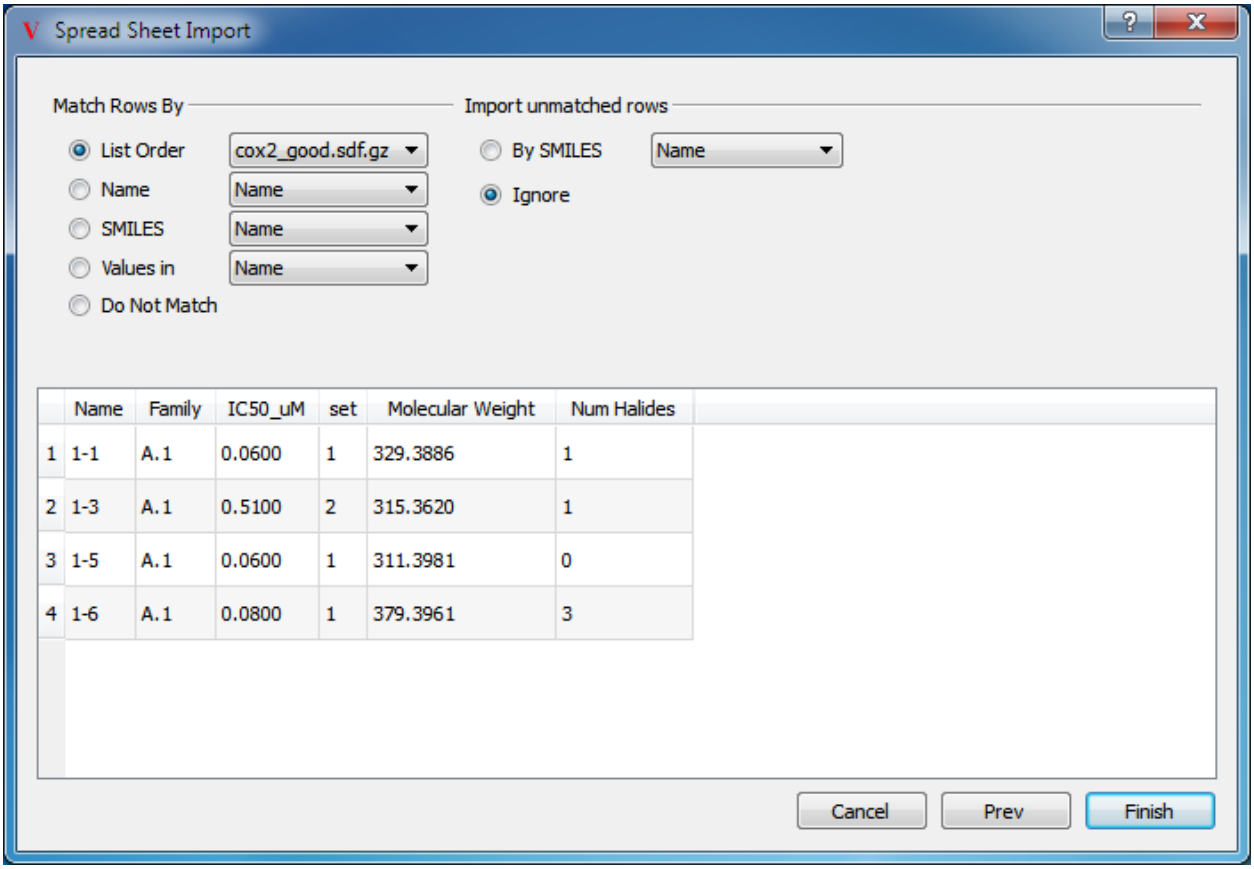


Fig. 15: **Importing (2)**
Specifying how to match the imported data

There are a number of mechanisms available to match data rows to the appropriate molecules in VIDA:

- List Order - matches the order of rows in the spreadsheet with the order of molecules in the selected list.
- Name - matches the molecule's name with the name in the selected column of the imported data.
- SMILES - matches the molecule's canonical SMILES string with the SMILES string in the selected column of the imported data.
- Values In - matches the value in the selected column between the imported data and the value already in the spreadsheet.
- Do Not Match - assume that you are importing new molecules

Unmatched rows can also be imported into the spreadsheet by creating a new molecule in VIDA to be associated with that row. The following options are available for unmatched rows:

- By SMILES - use the SMILES string in the selected column of the imported data to create a new molecule.
- Ignore - Do not import unmatched data.

6.10.2 Exporting

Spreadsheets may be exported to an external file by selecting the "Spreadsheet…" option in the *Export* submenu of the top level *File* menu. Selecting this option will launch a dialog which allows for specification of which columns to export (see *Figure: Exporting*). Supported formats include comma separated (.csv), tab delimited (.txt), and Portable Document Format (.pdf).

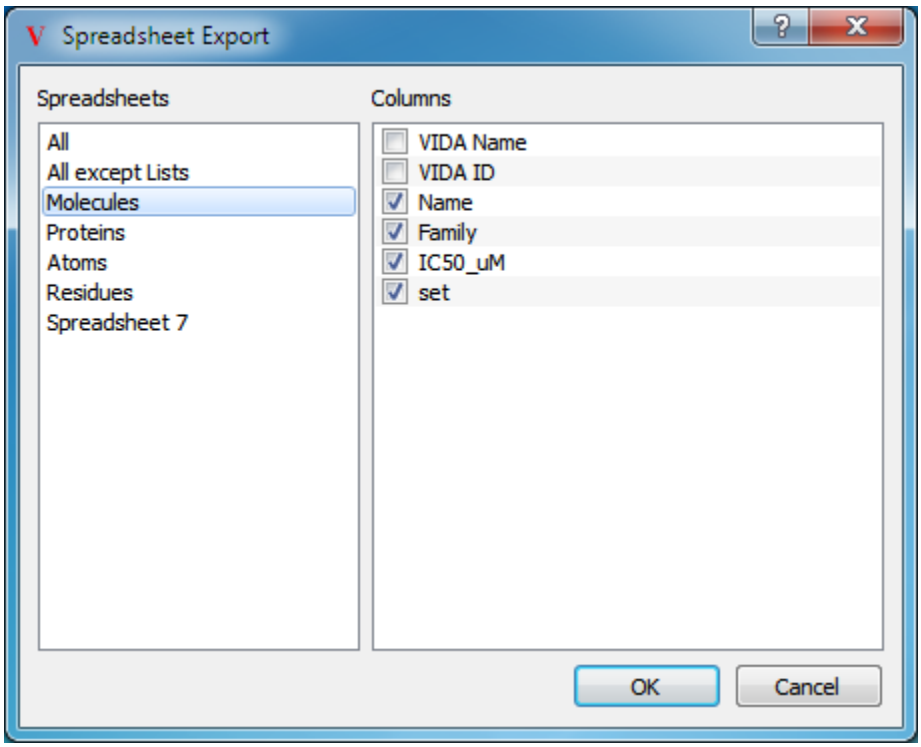


Fig. 16: **Exporting**
Specifying which spreadsheet and columns to export

6.11 Atom Spreadsheet

The *Atom* spreadsheet is a specialized spreadsheet provided by default that contains an entry for every atom in the *Active* molecule. A wide variety of atom properties can be displayed in the spreadsheet. Highlighting one or more rows in the spreadsheet will select the associated atoms in the 3D display. Atom visibility and marked status can also be controlled by toggling the state of the relevant columns.

6.12 Residue Spreadsheet

The *Residue* spreadsheet is a specialized spreadsheet provided by default that contains an entry for every residue in the *Active* molecule. Alternate residue conformations (if present) can be controlled by clicking on the relevant columns in the spreadsheet.

Note

In order to perceive alternate conformations the **ALL** input flavor must be turned on for the PDB file format. This can be set in the Molecule I/O section of the *application preferences*.

CHAPTER

SEVEN

LIST MANAGEMENT

The organization of molecules into lists and the ability to easily browse through these lists is an extremely important and central concept to VIDA. As such a List Window is provided to aid in the navigation and organization of lists. Furthermore, there exist many mechanisms to create, modify, and delete lists which will be discussed in more detail later in this chapter.

7.1 Introduction

The List Window is one of the primary navigation and organization mechanisms in VIDA. It displays all of the currently loaded files and their contents (molecules, grids, surfaces, etc.) into hierarchical lists for easy browsing and organization.

The List Window can be seen in *Figure: List Window* and contains multiple columns for controlling the various states and properties of the associated objects. The first column displays the name. The second column (with the green sphere icon at the top) is the *Visible* state column . The third column (with the padlock icon at the top) is the *Locked* state column. The fourth column (with the flag icon at the top) is the *Marked* state column. The fifth column contains the unique identifier assigned to that object. The sixth and last column contains the number of entries that are contained below that entry (e.g. number of molecules in a list, number of conformers in a molecule, etc). More details on the List Window columns are provided in the *List Columns* section.

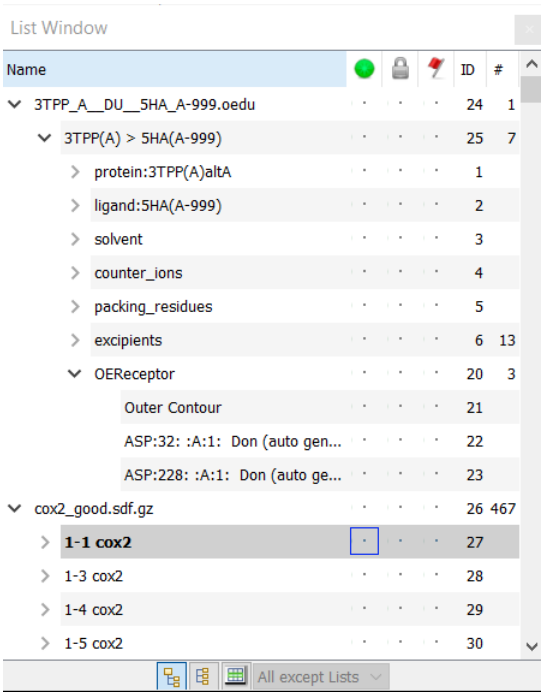


Fig. 1: **List Window**

7.2 Browsing

Every list in the List Window, as well as many of the other objects, can be expanded in the List Window to reveal further detail or the members contained within them. A small box with a "+" sign inside to the left of an object's name indicates that that object can be expanded. Molecules, for example, may be expanded to reveal conformers, molecular structure, or potentially other related children objects such as surfaces or grids. A single mouse click on the box will expand that object to reveal its contents. Double-clicking on the object name will turn on in-place renaming of that object.

7.3 Views

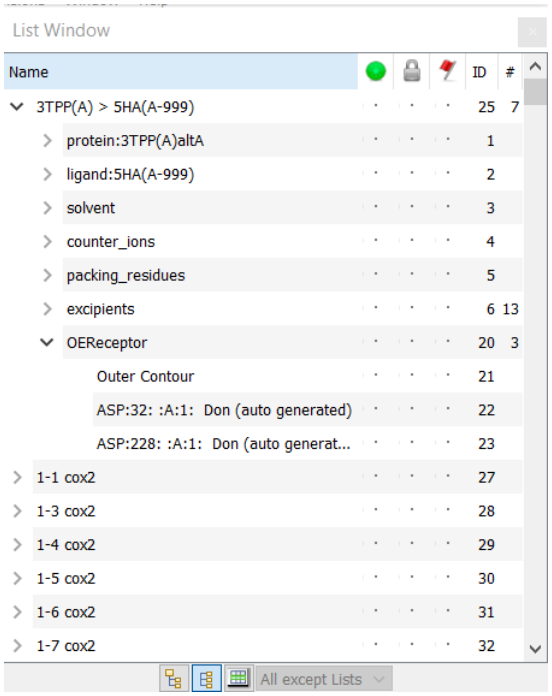


Fig. 2: **Flat View**

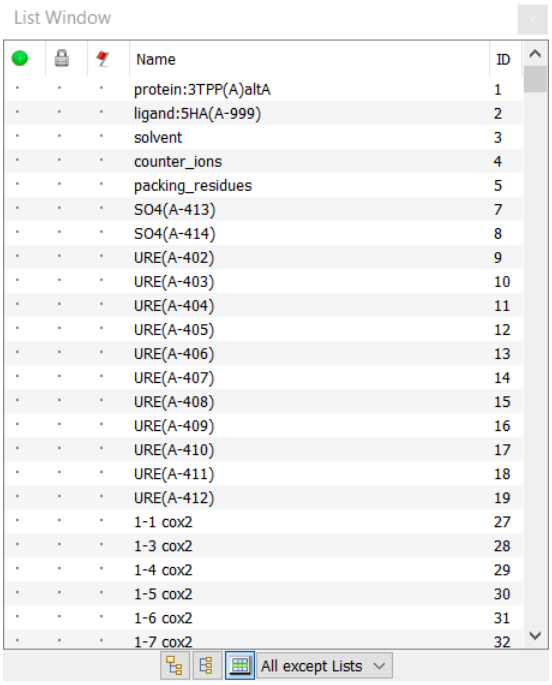


Fig. 3: **Spreadsheet View**

The List Window provides three different views onto the loaded data. Which view is shown is controlled by the three buttons at the bottom of the List Window. The first view is a traditional hierarchical view, described above, and seen in *Figure: List Window*. The second view is a flat view (see *Figure: Flat View*) in which the list hierarchy has been removed, but all the contents are shown in the same original order. The third view is a spreadsheet view (see *Figure: Spreadsheet View*) which shows only those objects that appear within the associated spreadsheet (as specified by the drop down box next to its button). Furthermore, the order of the objects in this view parallels the current sorting order in the application spreadsheet.

7.4 List Columns

There are multiple columns available for each object in the list. Some columns are simply informative, such as the unique identifier, while others are functional as well as informative.

Clicking on an object's name will make that object the *Focused* object. In the List Window, the *Focused* object's name is displayed in bolded text and there will be a small blue box around the dot in the second column (the *Visible* column). An example of this can be seen in *Figure: List Window*, where the molecule "1-1 cox2" is *Focused*.

The second column is the *Visible* column. The *Visible* column has a green dot icon in its column header. This column displays and controls the *Visible* property for objects. If an object is *Visible* there will be a green dot identical to the one in the column header present in that object's *Visible* column. If that object is also the *Focused* object, there will be a small blue box around the dot in that column as well. Clicking on the icon in the header of this column will make all lists visible.

If a given object has children, the visibility of its children can also be controlled in this column. Child objects have an additional layer of control regarding visibility in that they can be hidden, visible, or can mimic its parent's visibility. If an object is set to mimic's its parents visibility, an upright triangle is shown instead of the standard green dot (as can be seen in *Figure: Mimic Visibility*. Clicking in this column cycles through these three states in the following order: hidden, mimic, visible. Please note that molecular structure (atoms, residues, chains, etc) are not considered children objects and as such always mimic the parent molecule's visibility if they are marked visible with a green dot.

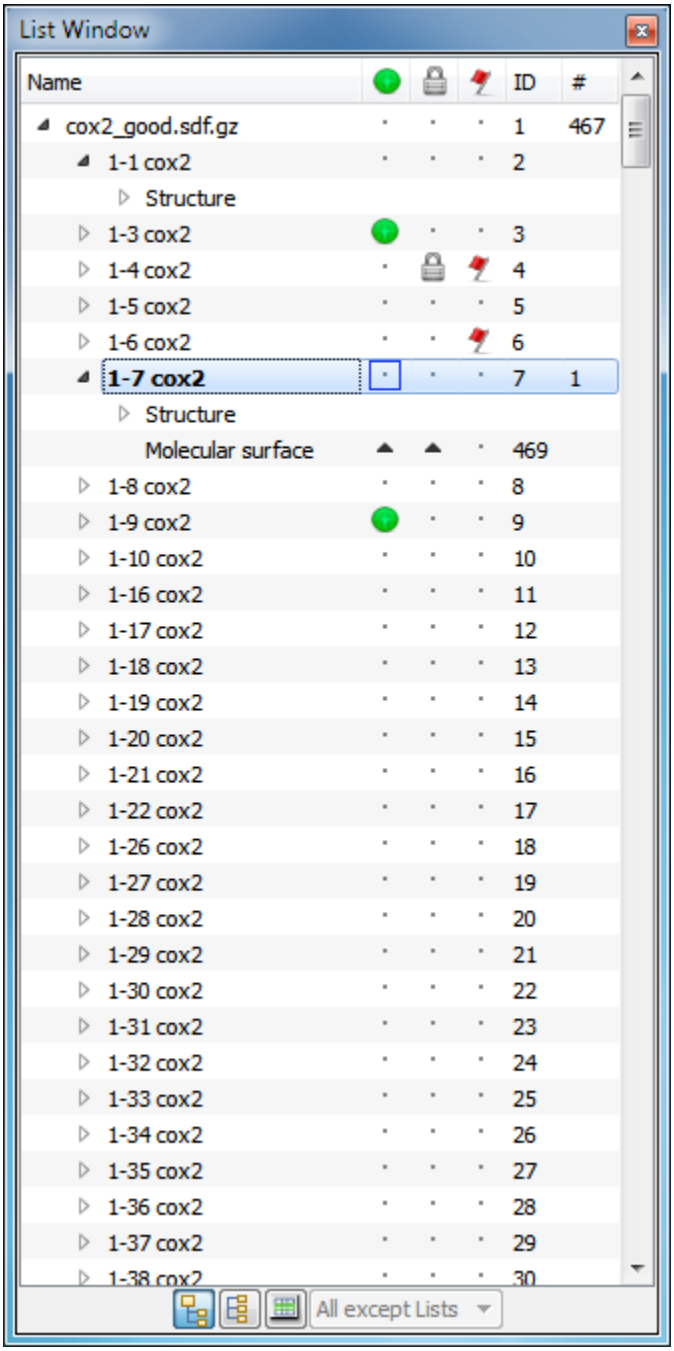


Fig. 4: **Mimic Visibility**

The third column is the *Locked* column. The *Locked* column has a small, grey padlock icon in its header. If an object is *Locked* there will be a grey padlock identical to the one in the column header present in that object's *Locked* column. Clicking on the lock icon in this column's header will lock all objects.

The fourth column is the *Marked* column. The *Marked* column has a red flag icon in its header. If an object is *Marked* there will be a red flag identical to the one in the column header present in that object's *Marked* column. Clicking on the flag icon in this column's header will mark all objects.

The fifth column is the *ID* column. An object's ID is simply a unique identifier that can be useful to identify specific

objects when running various commands. Each top level object has its own unique ID, but child objects like conformers and atoms are all a part of their parent molecule's ID.

The sixth and final column is the *Number* column. This column contains the number of entries available beneath the current object. For lists, this will display the number of molecules in that list. For molecules, this will display the number of conformers (if there is more than 1).

Columns can be shown or hidden as desired by right-clicking in the header of the List Window. These changes will be remembered in the *Preferences* and thus persist between sessions.

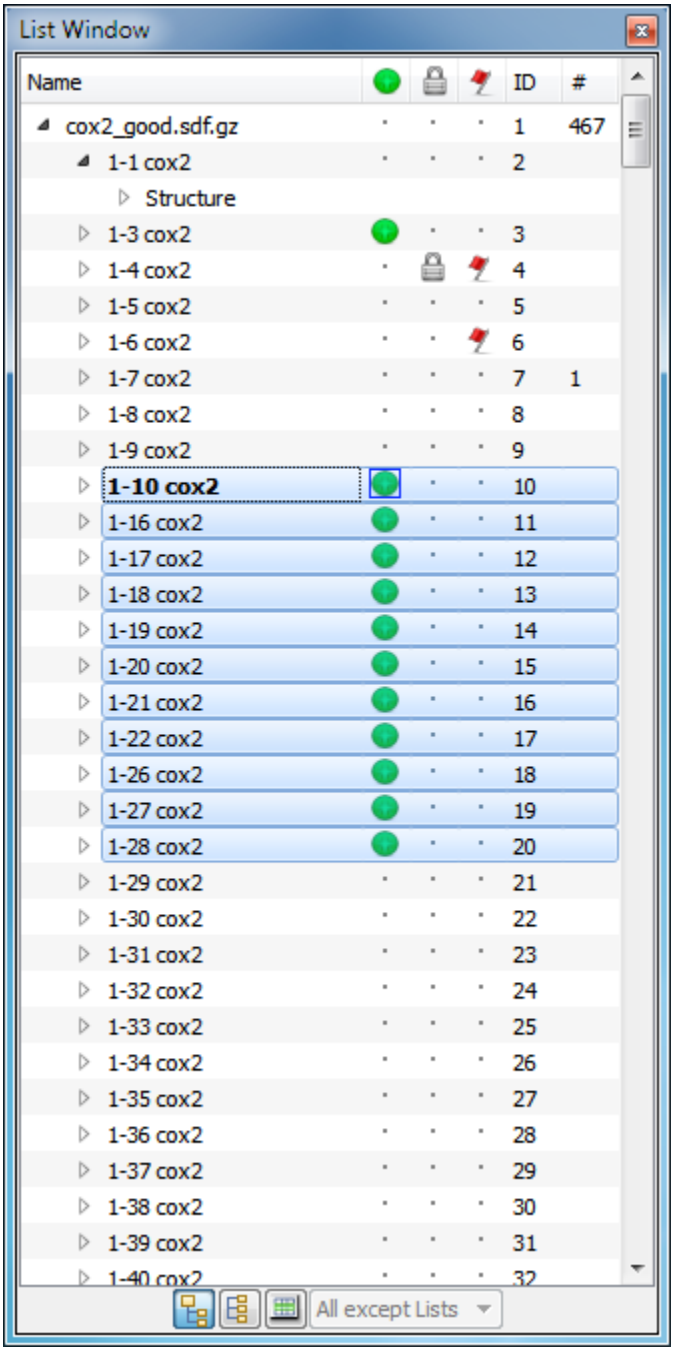


Fig. 5: **Multiple Objects**

The *Visible*, *Locked*, and *Marked* properties can be toggled by clicking on their associated columns in the List Window. If multiple objects are highlighted simultaneously and the click occurs in a column associated with the highlighted region, that change will apply to all of the highlighted objects. In *Figure: Multiple Objects*, the highlighted objects have been made *Visible* by clicking in that column. The *Focused* object has changed as a result as well.

Visible, *Locked*, and *Marked* properties of the *Focused* object can be toggled using the V, L, and M keys, respectively.

7.5 List Manipulation

The List Window operates on the basic concept that every potentially viewable object is placed inside a list. As a result, the root items in the List Window are always lists. These lists can be combined, subsetted, and manipulated into new lists. The properties of all the objects in a list can be set together by clicking in the columns of their containing list. Finally, objects can easily be removed from and inserted into lists.

7.5.1 Creating Lists

New lists can be created in several ways. The simplest way is to select some number of molecules from a list and drag them to the top of the List Window.

The "New List" submenu in the *File* menu provides a number of list creation methods, including using the AND of two or more lists (intersection), the OR of two or more lists (union), the XOR of two lists (union - intersection), or from the currently *Marked* set of objects.

The AND list contains only those objects which are in all of the source lists. The OR list contains all of the objects contained in its individual source lists. The XOR list contains the objects which are in only one of the source lists, but not both. Finally, the *Marked* list creates a new list which contains all of the currently *Marked* objects.

Another way to create a new list is by right-clicking on an individual list and selecting the "Subset" submenu from the popup menu. This menu contains four entries – "By Marked", "By Query", "By Not Marked" and "By Not Query". The "By Marked" option creates a new list containing all of the *Marked* objects from within the list. The "By Not Marked" option creates a list with all of the objects from the list which are not *Marked*. "By Query" generates a dialog which prompts for a substructure query that if the objects match, those objects will be placed in the new list. The "By Not Query" behaves the same way, except that it puts the objects that do match the query into the new list.

The last way to create a new list is from the "Find" menu entry in the *Edit* menu. This mechanism searches all of the currently loaded molecules by either Amino Acid Sequence, SMARTS, a regular expression against the molecule title, or by a Query molecule (as a substructure search). After clicking the "Search" button, the matching results will be displayed on the right-hand side of the dialog. The entire set or just specific individual results can be selected within the dialog to populate the new list.

7.5.2 Reorganizing Lists

The List Window supports "drag and drop" of lists and the items they contain. This allows lists to be reordered, molecules to be reordered within a list, and molecules to be dragged from one list to another. New lists can be created by dragging molecules out of a list to the List Window's top level.

> **Note**
>
> OEDesignUnit entries cannot be edited at this time

7.5.3 Deleting from Lists

As previously mentioned, objects can also be removed from lists. There are two ways to remove an object from a list. The first mechanism is to delete the object (by selecting the "Delete" option from the right-click menu). Deleting an object will remove it from ALL of the lists to which it belongs. Choosing the "Remove from List" option from the same right-click menu will remove the object from the specified list only, leaving it perfectly intact in any other lists that it might belong to. When the last object is removed from a list, the list itself will be removed.

CHAPTER

EIGHT

PROCESSES

The Process Manager allows the user to start, monitor, terminate, and load results from processes external to VIDA. As part of this functionality, it is possible to write out currently loaded molecules (or grids) to files which can be used as input to the external processes. Furthermore, it is possible to create "pipes" which chain together the input and output of individual processes into a larger process.

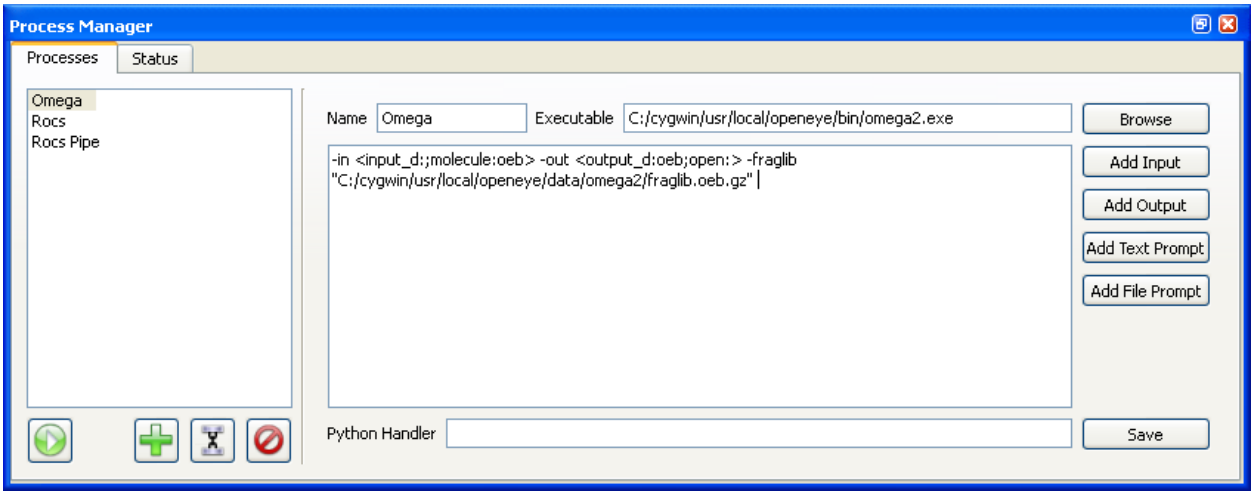


Fig. 1: *Process Manager*

The Process Manager contains two tabbed windows called "Processes" and "Status" respectively as can be seen in Figures X.Y and X.Y. The "Processes" tab contains the list of available processes and the functionality to create and edit them. The "Status" tab displays all of the currently active and recently completed processes and information about their state. It also displays the commandline output generated by each of the processes.

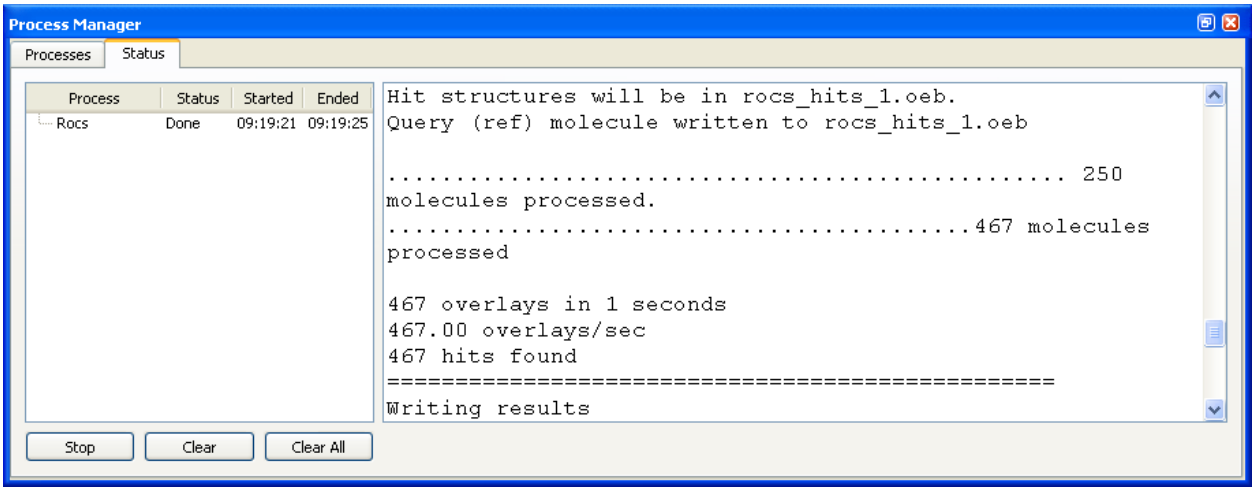


Fig. 2: *Output generated by a completed process*

Beneath the list of processes in the "Processes" tab is a collection of four buttons. The button on the left with the play icon starts the currently selected process. The next button with the plus icon clears the editor on the right hand side to allow for creation of a new process. The next button with the link icon changes the editor on the right hand side to allow for creation of a new pipe (see Figure X.Y). The last button with the delete icon deletes the current selected set of processes from the list of available processes.

8.1 Processes

When creating a new process or editing a current one, there are a number of available options. At the top of the editor there is a field for the process name (which will be displayed in the list of available processes seen on the left as described above) as well as a field for the actual location of the application to be run. The "Browse" button to the right of this field can be used to find the location of the desired application.

Beneath this is a large text entry box which allows for specification of any desired command line arguments to the application. It is not necessary to enter the application name here.

For most applications, at least one filename will have to be specified as a command line argument. This can be done multiple ways. If the file to be specified is always going to be located in the same location, the filename can be entered directly into the text entry box as a command line argument. If the location of the file is not likely to be constant, a prompt can be specified as opposed to an actual filename. To add a prompt, click on the "Add File Prompt" button to the right of text entry box. If the desired file will need to be created each time, it is possible to create the desired file at start-time by writing out selected molecules (or grids) directly from VIDA. To use this feature, click on the "Add Input" button to the right of the text entry box. At start-time, the user will be prompted to select the desired molecules (or grids), which will then be saved to a file which will be passed as a command line argument to the process.

In addition to being able to specify specific input files, it is possible to specify an output file which can potentially be read back into VIDA when the process is completed. As discussed above, it is possible to a specify specific filename in the command line or to use a prompt, but if it desired to load the resulting file, the "Add Output" button should be clicked to specify the file.

Besides being able to specify filenames, it is often desirable to be able to specify other variable arguments at start-time. To do this, click on the "Add Text Prompt" button which will add a prompt argument to the command line arguments. At start-time the user will be asked to specify the desired text to be passed to the process.

Lastly, at the bottom of the Process Editor is a "Python Handler" field which allows for specification of a Python function to be run when the process is completed. This allows for handling of output not normally handled by VIDA or for output which is not specified on the command line.

8.1.1 Adding Input

The Input Editor allows the user to specify a mechanism by which an input file is provided to the process to be run. At the top of the editor are two mutually exclusive options, the first is the "Temporary" check box which indicates that the file to be used will be a temporary file which will be removed after the process has completed. The other option is a field which allows specification of a specific filename to be used. In both cases, the user has a choice as to how this file will be populated: saving molecules, saving grids, or running a specified Python function which will handle the population of the file. If molecules or grids are chosen, a file format to save these in must be specified in the pull-down box next to the selected option.

If a Python function is specified to populate the input file, the desired filename can be obtained by using the keyword "@FILE@" in the function specification. This keyword will be substituted with the real filename at start-time before the Python function is called.

Finally, there is a checkbox at the bottom of the dialog to specify whether or not this is a default input. The default input entry is used when constructing pipes.

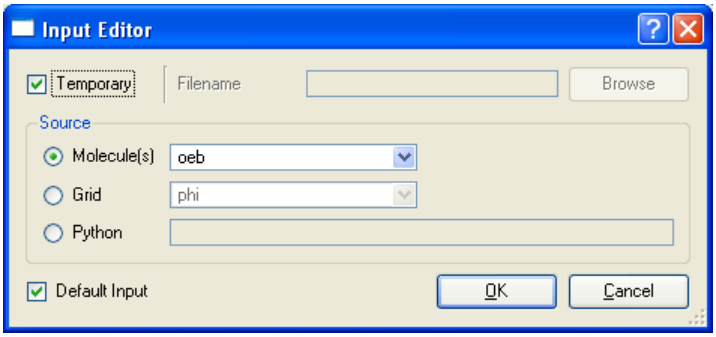


Fig. 3: *Process Input Editor*

8.1.2 Adding Output

The Output Editor allows the user to specify a mechanism by which an output file of the process to be run is handled by VIDA. At the top of the dialog are two mutually exclusive options, the first is the "Temporary" check box which indicates that the output file is a temporary file and will be removed after it has been handled by VIDA. The other option is a field which allows specification of a specific filename to be used for the output file. If this option is selected, the file will not be deleted after it is handled.

Once the above has been specified, the user can specify how VIDA should handle the file when the process is finished. There are three possible options, the first is to open the file, the second is to ignore the file, and the last is to call a Python function to handle the resulting file. If it is desired to open the file, the file format that the process should use to generate the output file must be specified so that VIDA will know how to properly load the results.

If a Python function is specified to handle the output file, the actual filename can be obtained by using the keyword "@FILE@" in the function specification. This keyword will be substituted with the real filename at process completion time before the Python function is called.

Finally, there is a checkbox at the bottom to specify whether or not this is a default output. The default output entry is used when constructing pipes.

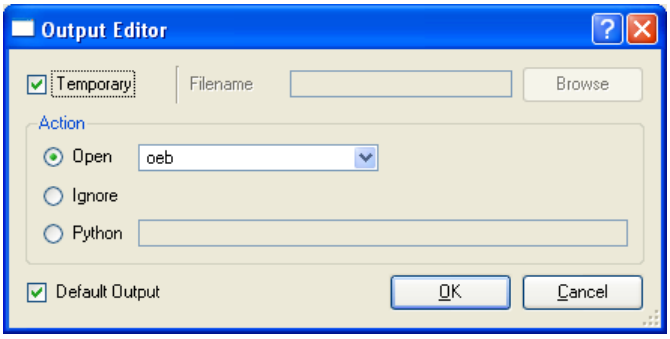
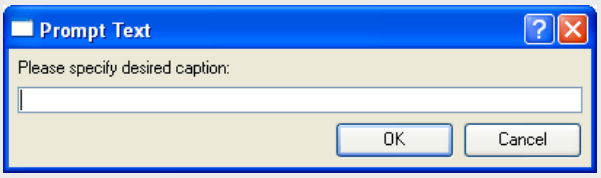


Fig. 4: *Process Output Editor*

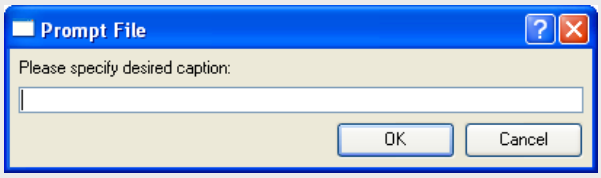
8.1.3 Adding Prompts

As described previously there are two types of prompts that are available for specification on the command line: a text prompt and a file prompt.

The Text Prompt Editor (see Figure X.Y(a)) allows the user to add a text prompt to the process as well as to specify what the caption of the prompt will be. Specification of an informative caption is helpful in allowing the user to know what information is being requested.



Text Prompt Editor



File Prompt Editor

The File Prompt Editor (see Figure X.Y(b)) allows the user to add a file prompt to the process as well as to specify what the caption of the resulting prompt will be. Specification of an informative caption is helpful in allowing the user to know what specific file or type of is being requested.

The result of both of these prompts are passed as command line arguments to the process in the order in which they appear in the main text entry box in the Process Editor (see Figure X.Y).

8.2 Pipes

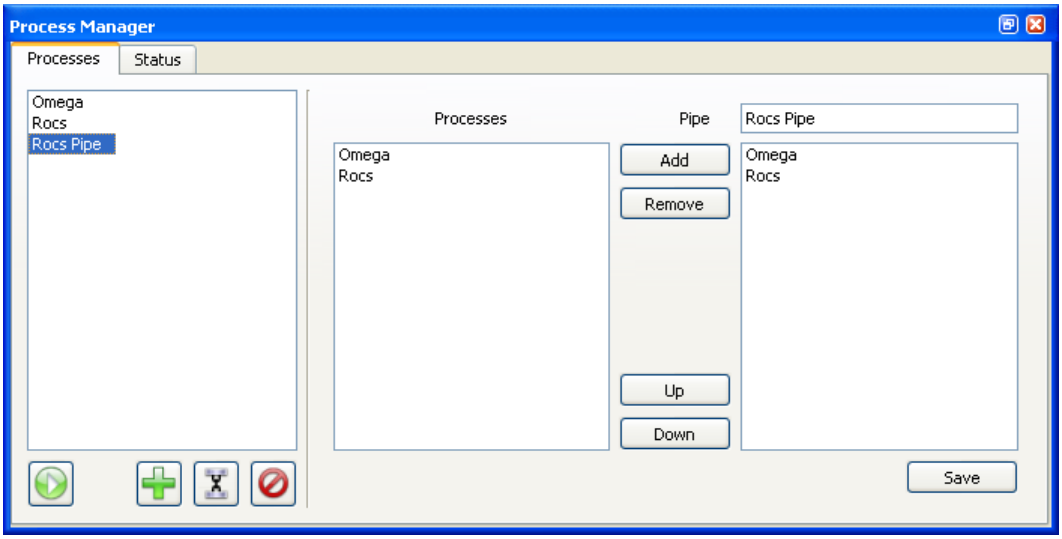


Fig. 5: *Pipe Editor*

Once a number of processes have been created, they can be "piped" together using the Pipe Editor (see Figure X.Y). On the left-hand side of the editor is a list of all the currently defined processes. On the right is an ordered list of processes to be run and in which order to run (top to bottom).

When chaining processes together into a pipe, it is important that they have default input and output fields specified. The default output of a given process is passed into the default input of the following process. When the final process is completed, the user will be notified and the final output file will be handled based on its specification in the associated process.

CHAPTER

NINE

MOLECULE MANIPULATION

VIDA has the ability to both create and modify existing molecules. The focus is primarily on small molecules, but it is not limited to small molecules. Molecules can only be modified in a specific *Editing* mode, more details of which are available in the following sections.

9.1 Creating Molecules

A new molecule can be created from scratch in VIDA by clicking on the button in the main toolbar with the icon of a molecule overlaid with a plus sign (see *Table: Builder Icons*). Clicking this button will open VIDA's sketcher window.

9.1.1 VIDA Sketcher Window

The 2D sketcher window contains a basic 2D sketching environment at the top and a 1D text entry mechanism below that. The 1D input mechanism provides a very fast way to specify a new molecule or get started with new molecule creation. The 1D input mechanism supports SMILES, IUPAC and common names, as well as FASTA sequences for small peptides (sequence must be prefixed by a leading angle bracket (>) character). Disconnected fragments can be specified on the 1D input line by entering a vertical bar (|) character between the entries. Once a valid molecule has been specified in the 1D area, that molecule will be depicted in the 2D sketching area.

A basic overview of the 2D sketcher can be seen in Figure: 2D Sketcher

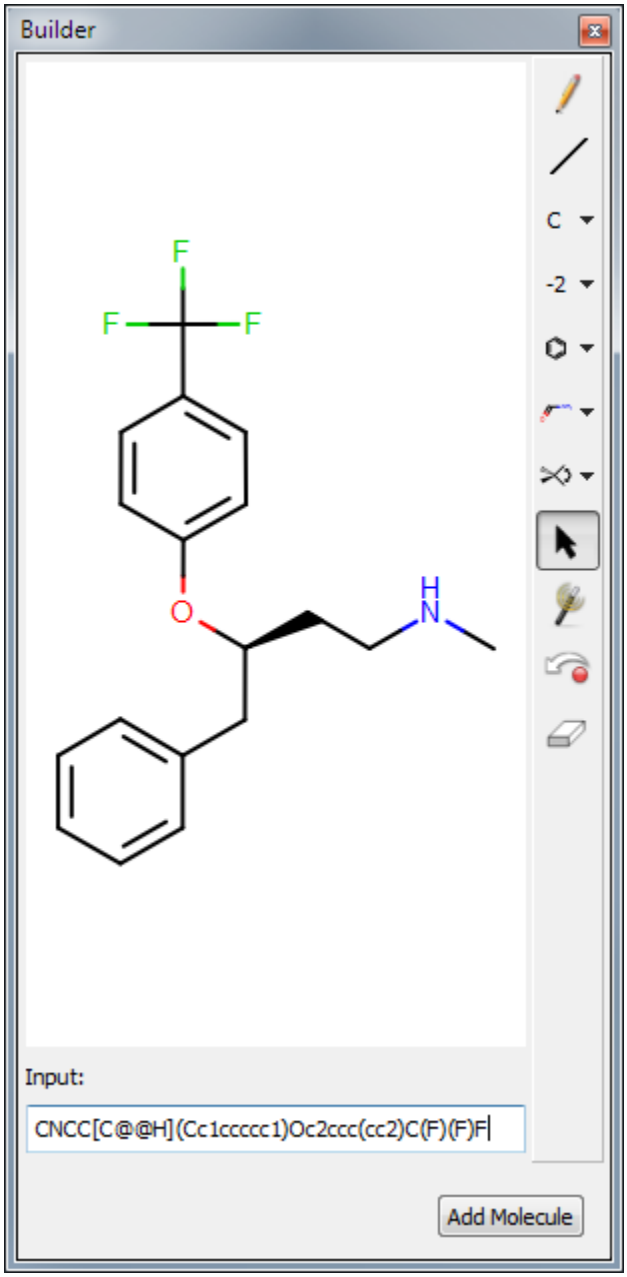


Fig. 1: **2D Sketcher**

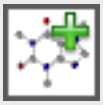
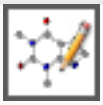
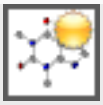
Icon	Description
	**Pencil:** This is the default drawing tool. With the pencil icon selected (highlighted) click in the main sketcher window to start drawing a new molecule. Clicking on an atom in the main sketcher window will incrementally build up a carbon chain. The attachment points are highlighted in orange. Clicking on a bond will toggle the bond order. This tool can also be used to close rings. Click on the first atom and drag to the second atom where the ring will close. Hovering the mouse over an atom or bond highlights it orange. Right-click on the highlighted atom to access a right-click menu which allows the user to *Set Element*, *Set Formal Charge* or *Delete* the selected atom/bond. If a bond is highlighted then the right-click menu offers an option to *Set Bond Order*. There are also options to *Attach…* (through an atom) or *Fuse…* (through an atom or bond) a functional group. In these cases a secondary dialog opens in which a SMILES string can be input of the group to be attached or fused.
	**Bond order:** Clicking on a bond in the main sketcher window will cycle through the available bond orders for that bond. Bond order can also be set through the *Set Bond Order* option of the right-click menu. Hovering the mouse over an individual bond will temporarily highlight it orange and the right-click menu will operate on that bond only. If multiple bonds are selected (and therefore highlighted orange) using the *Selection* tool (below) then they will ALL be set to the new bond order using *Bond Order* in the right-click menu. Available options are 0-4 where 0 will break the bond.
	**Element:** Click on the small arrow to open a dropdown menu and select from some commonly used elements. Clicking on an atom in the main 2D sketcher window will cause that atom to be changed to the selected atom type. In addition to the commonly used atoms there is an "X" dummy atom type. Clicking the "…" option in the list opens a full periodic table of the elements from which any element can be selected. This icon shows an image of the currently active atom type. The default is "C". Atoms types can also be set through the *Set Element* option of the right-click menu. Hovering the mouse over a single atom will temporarily highlight it orange and the right-click menu will operate on that atom only. If multiple atoms are selected (and therefore highlighted orange) using the *Selection* tool (below) then they will ALL be set to the same atom type using *Set Element* in the right-click menu.
	**Charge:** Click on the small arrow to open a dropdown and select the formal charge for an atom. Options available are -2, -1, 0, +1, +2. With the charge icon active (highlighted white) clicking on any atom in the main sketcher window will modify the atom to have the currently selected charge. This icon displays an image of the currently active charge. The default is "-2". Formal charge can also be set through the *Set Formal Charge* option of the right-click menu. Hovering the mouse over an individual atom will temporarily highlight it orange and the right-click menu will operate on that atom only. If multiple atoms are selected (and therefore highlighted orange) using the *Selection* tool (below) then they will ALL be set to the new formal charge using *Set Formal Charge* in the right-click menu. This method provides a wider range of charge values than can be accessed using the *Charge* icon.
	**Rings:** Click on the small arrow to open a dropdown menu and select from some commonly used rings, such as benzene, cyclopropane and cyclohexane. The selected ring can be added to the growing molecule multiple times. The attachment atom or fusion bond is highlighted in orange and alternative fusion options can be accessed using the mouse scroll wheel. This icon shows an image of the currently active ring template. The default is benzene.

	**Templates:** Click on the small arrow to open a dropdown menu and select from numerous predefined template functional groups, linkers and rings. There is also an option to define a Custom functional group. This opens a new dialog into which a SMILES string can be input. The selected functional group, linker or ring can be added to the growing molecule multiple times. The attachment atom or fusion bond is highlighted in orange and alternative fusion options can be accessed using the mouse scroll wheel. This icon shows an image of the currently active functional group, linker or ring. The default is "C". With the default "C" functional group selected clicking on an atom in the main sketcher window will build up a carbon chain. This tool can also be used to close rings. Click on the first atom and drag to the second atom where the ring will close.
	**Stereochemistry:** Click on the small arrow to open a dropdown and select atom and bond CIP stereochemistry options. Available options are R and S for chiral atoms and E or Z for chiral bonds. There is also an option to invert the current stereochemistry. This icon displays an image of the currently active option. The default is "Invert". If no stereochemistry is already set then clicking an atom with the *Invert* option selected will initially set the stereochemistry to R and subsequent clicks with *Invert* will toggle between R and S. In the case of bonds where no stereochemistry is already set the *Invert* option will initially set the stereochemistry to E and then toggle between E and Z. Thus *Invert* can be considered a 'one stop shop'.
	**Select:** With the arrow icon highlighted drag to draw a loop around the atoms to be selected or Shift-click or Ctrl-click to incrementally select multiple atoms and/or bonds. The selected atoms and bonds will be highlighted in orange. A right-click menu allows the user to *Set Element*, *Set Formal Charge* or *Delete* the selected atom(s). This operation acts on ALL the selected atoms *e.g.* they would all get the same charge. If only bonds are selected then the right-click menu offers an option to *Set Bond Order*. There are also options to *Attach...* (through an atom) or *Fuse...* (through an atom or bond) a functional group. In these cases a secondary dialog opens in which a SMILES string can be input of the group to be attached or fused.
	**Clean up the display:** This tool is useful after portions of the molecule have been deleted, especially if a ring has been cleaved. It uses an OEDepict algorithm to redraw the molecule according to an internal set of rules.
	**Undo:** Clicking on this icon will undo the last action in the main sketcher window. Repetitive use will undo actions further back in the history. *Ctrl-Z* is a keyboard shortcut that can also be used.
	**Eraser:** Delete atoms and/or bonds by selecting the eraser tool and clicking on the atom or bond to be deleted. Atoms and bonds can also be deleted through the *Delete* option of the right-click menu. Hovering the mouse over a single atom or bond will temporarily highlight it orange and the right-click menu will operate on that atom/bond. If multiple atoms/bonds are selected (and therefore highlighted orange) using the *Selection* tool (below) then they will ALL be deleted using *Delete* in the right-click menu.

9.2 Editing Molecules

Molecules can only be changed in VIDA when in *Editing* mode. Only a single molecule can be in *Editing* mode at a time. To put a molecule into *Editing* mode, select the "Edit" option from the right-click popup menu or click on the button in the main toolbar with the icon of a molecule overlaid with a pencil (see *Table: Builder Icons*).

Table 1: **Builder Icons**

Button	Description
	**Sketch new molecule:** Opens the 2D Sketcher to allow creation of a new molecule.
	**Edit current molecule:** Places the current active molecule into the 3D Builder and 2D Sketcher.
	**Save as new molecule:** Saves the edited molecule as a new entry while remaining in Editing mode.
	**Accept edits:** Leaves Editing mode and keeps the changes made to the edited molecule.
	**Cancel edits:** Leaves Editing mode and cancels any unsaved edits.

On entering *Editing* mode, the "Open 3D Builder" icon will be replaced by one of a molecule with a green "+" sign. This icon will close the Builder and keep any changes made to the molecule during editing. An additional button also appears in the toolbar with the icon of a molecule overlaid with a red "X". Clicking on this button will cancel out all of the changes made to the molecule and will restore the molecule back to its original state.

When VIDA enters *Editing* mode, a number of builder tools appear in the 3D window, and the Sketcher window opens, allowing editing of a molecule in either 2D or 3D. The 2D and 3D editing functions are roughly equivalent, and are synchronized, so that selecting a tool in the 3D window will select the corresponding tool (where possible) in the 2D sketcher and vice versa. The various building tools are discussed in detail in *Creating Molecules*.

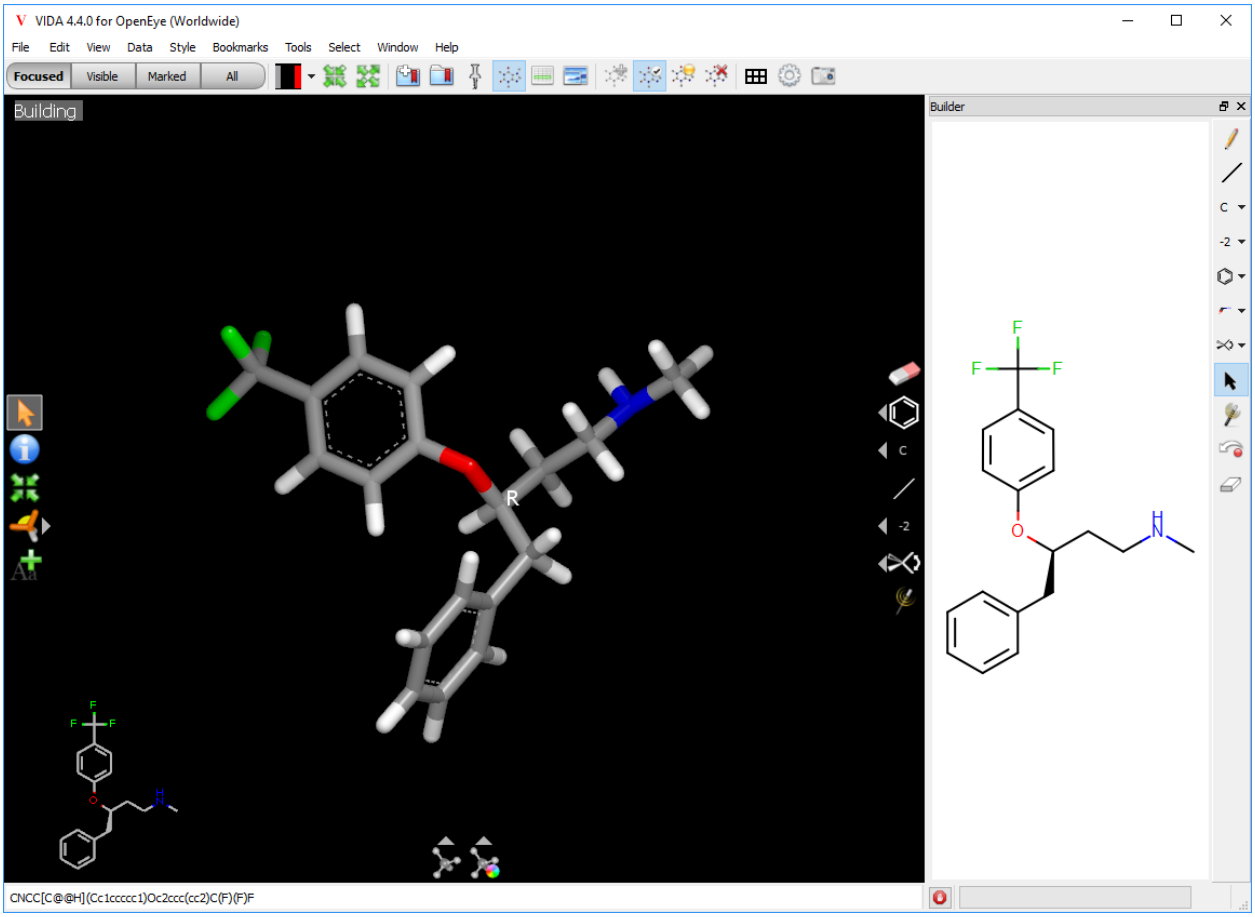


Fig. 2: **Molecule Editing**

Icon	Description
	**Select:** This is the default selection mode. With the icon highlighted click on the atoms or bonds to be selected or Shift-click or Ctrl-click to incrementally select multiple atoms and/or bonds. The selected atoms and bonds will be highlighted in orange. Double click to select whole functional groups or molecules. Select a group of atoms and bonds within a rectangle using right-click and drag.
	**Information:** Clicking on an atom or bond provides information such as atom type, 3D Cartesian coordinates, bond type and bond length.
	**Center:** Select the center tool and then click on an atom to center the view on that atom
	**Monitors:** Place monitors for distance (2 atoms), angle (3 atoms) or torsion (4 atoms). Click sequentially on the atoms to be included in the monitor. Before the last atom is clicked hovering over an atom will display a temporary monitor. Clicking on the atom makes the monitor persist.
	**Add Labels:** With the tool selected click on an atom or bond to add label text.
	**Delete:** Delete atoms and/or bonds by selecting the eraser tool and clicking on the atom or bond to be deleted. To delete groups of atoms or bonds first select them with the Selection tool and then either use the *Delete* option in the right-click menu or use the *Delete* keyboard key.
	**Rings:** Click on the small arrow to open a dropdown menu and select from some commonly used rings, such as benzene, cyclopropane and cyclohexane. The selected ring can be added to the growing molecule multiple times. There is also an option to define a Custom functional group. This opens the 2D sketcher into which the functional group can be sketched or a SMILES string can be input. This icon shows an image of the currently active ring template. The default is benzene.
	**Element:** Click on the small arrow to open a dropdown menu and select from some commonly used elements. Clicking on an atom in the main 3D window will cause that atom to be changed to the selected atom type. In addition to the commonly used atoms there is an "X" dummy atom type. Clicking the "…" option in the list opens a full periodic table of the elements from which a wider range of elements can be selected. This icon shows an image of the currently active atom type. The default is "C".
	**Bond order:** Clicking on a bond in the main 3D Builder window will cycle through the available bond orders for that bond.
	**Charge:** Click on the small arrow to open a dropdown and select the formal charge for an atom. Options available are -2, -1, 0, +1, +2. Other formal charges not in this range can be set with the "…" option. With the charge icon active (highlighted white) clicking on any atom in the main sketcher window will modify the atom to have the currently selected charge. This icon displays an image of the currently active charge. The default is "-2".
	**Stereochemistry:** Click on the small arrow to open a dropdown and select atom and bond CIP stereochemistry options. Available options are R and S for chiral atoms and E or Z for chiral bonds. There is also an option to invert the current stereochemistry.

When a molecule enters *Editing* mode, VIDA adds explicit hydrogens to the molecule to properly fill out the valences. In addition, labels are added to both atoms and bonds to indicate stereochemistry and formal charge.

A molecule can be edited by either setting the mouse mode using the buttons on the right or by clicking on one or more atoms/bonds and then selecting the desired operation from the context sensitive right-click menu. The menu provides the following choices. Some options are available only when atoms are selected and others only when bonds are selected.

Table 2: **Builder Right-Click Menu**

Center	Centers the highlighted atom(s) in the display.
Set Element	Atoms only. Select the desired atom type from a Periodic Table. This atom type will be applied to all the selected atoms.
Set Formal Charge	Atoms only. Select the formal charge for all the selected atoms.
Set Torsion	Specify a torsion angle for a 4-atom set.
Set Bond Order	Bonds only. Specify the desired bond order. Applies to all the highlighted bonds.
Set Bond Length	Bonds only. Specify the desired bond length. The new length is applied to all the highlighted bonds.
Modify Torsion	Bonds or 4-atom sets only. Use the keyboard *up* and *down* arrow keys or the mouse scroll wheel to increase and decrease the bond torsion.
Modify Angle	3-atom sets only. Use the keyboard *up* and *down* arrow keys to increase and decrease the bond angle.
Sprout Methyl	Atoms only. If the selected atom has an open valence (i.e. bond to H) this option will replace the H-atom with a methyl group.
Create Bond	Atoms only. Make a new bond between two selected atoms with open valences.
Attach	Atoms only. Opens the 2D sketcher to input the fragment to be attached to the highlighted atom.
Fuse	Opens the 2D sketcher to input the fragment to be fused to the highlighted atom or bond.
Measure Length	Bonds only. Displays a monitor with the bond length.
Measure Torsion	Bonds only. Displays a monitor for the torsion around the bond(s).
Measure Angle	Atoms only. Displays a monitor with the bond length.
Minimize	Atoms only. Minimize the highlighted portion of the molecule using the MMFF94 or MMFF94s force field.
Delete	Delete the highlighted atoms or bonds. Open valences will be filled with H-atoms.
Cancel Editing	Exit the 3D builder without keeping any of the changes.
Finish Editing	Exit the 3D builder and accept the edits.
Create	A sub-menu provides options for creating molecular or accessible surfaces or electrostatic grids.
Bookmarks	A sub-menu provides options to add or organize bookmarks and define animation preferences.
Delete Visible Monitors	Deletes all visible monitors (angle, length, torsion) from the 3D display.

9.2.1 Independent Rotation/Translation of Molecules

It is possible to rotate and translate a molecule in *Editing* mode independently from all other molecules in VIDA, by holding down the *Ctrl* key (*Command* key on the Macintosh) during rotation or translation actions. Any selected atoms will define the center of rotation. If no atoms are selected, the centroid of the molecule is used for the center of rotation.

9.2.2 Minimization

An entire molecule or just a subset of a molecule can be minimized using the MMFF94 or MMFF94s forcefield. This minimization takes place in a vacuum and does not take into account any context (such as a protein active site).



Fig. 3: **Minimize Tool**

To minimize a molecule, the molecule of interest must be in *Editing* mode. Once *Editing* mode has been enabled for a molecule, a button with a magic wand icon (see *Figure: Minimize*) will appear on the right hand side of the 3D display. Clicking on this button will cause the entire molecule or just the selected subset of the molecule to be minimized. Like all other procedures in VIDA, the minimization process can be undone.

9.2.3 Monitors

Angle, distance, and torsion monitors can be added to individual molecules to display as well as modify the associated properties. Monitors can be created using the monitor mouse mode (available from the toolbar on the left side of the *3D display*). Torsion monitors can also be created by first selecting a single bond, three adjacent bonds, or four adjacent atoms and then choosing the *Modify Torsion* option in the right-click menu.

The value associated with a distance or angle monitor can be changed by selecting the monitor of interest and then scrolling the mouse wheel. By default, the smaller side of the molecule will be the one modified by these changes; however, if the *Ctrl* key is held down while using the mouse wheel, the larger side will be modified instead.

In the builder, torsion monitors are added in pairs, with one monitor at each end of the selected bond. The monitor you select will determine which end of the bond will rotate. (By default the monitor on the smaller side of the molecule is selected.)

To rotate torsions within a ring, create a torsion monitor on a ring bond, then select one other bond in the ring as a "closure bond". The closure bond will stretch and deform as you adjust the torsion. When finished with the adjustment, you will probably want to minimize the ring structure using the *Minimize Tool* (see *Figure: Minimize*).

The figure below illustrates adjusting a ring torsion to convert a cyclohexyl group from chair to boat conformation. The active torsion monitor and closure bond are shown selected.

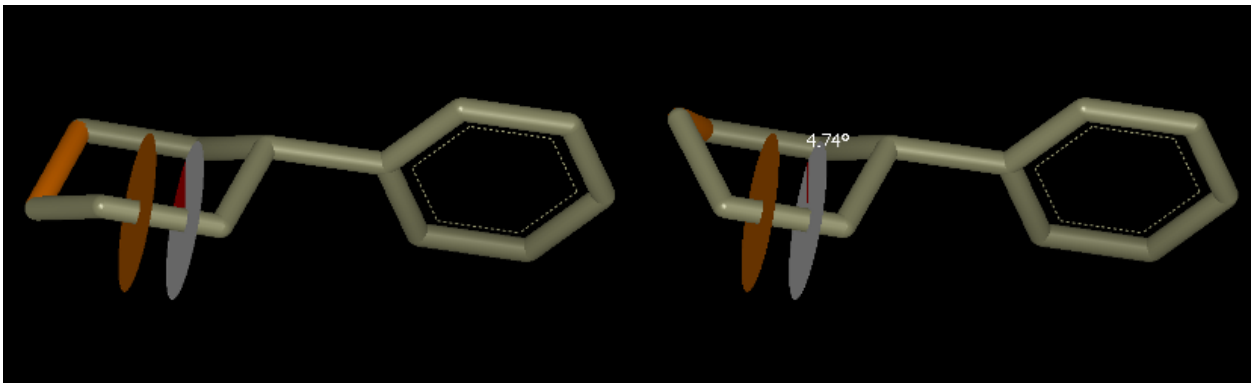


Fig. 4: **Ring Pucker Conversion.** *Interconversion of a ring from chair to boat conformation can be performed by rotating a single ring torsion.*

Torsion monitors can also be used to convert between equatorial and axial substitutions of a ring, as shown in the following figure. In this example, a single torsion is rotated in two steps, using two different closure bonds.

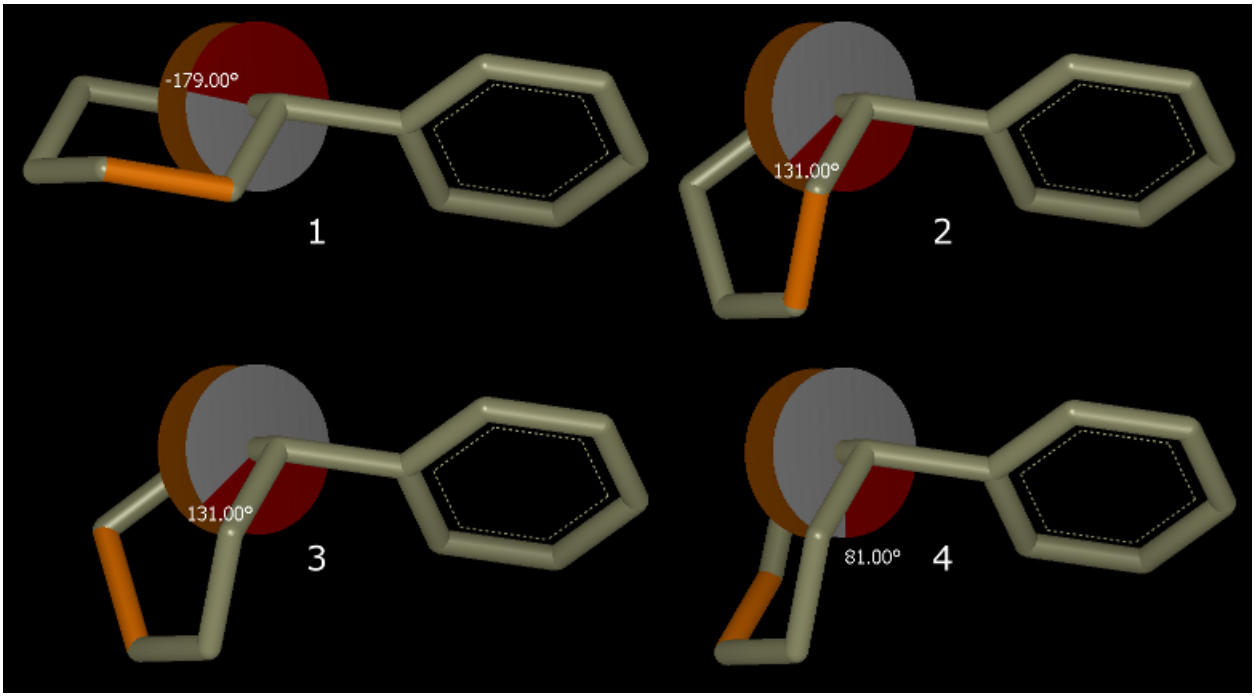


Fig. 5: **Equatorial to Axial Conversion.** *An equatorial substitution on a cyclohexyl ring is converted to an axial substitution by adjusting a ring torsion. In step 3 a new closure bond is selected, and the torsion rotated farther, until a chair conformation is obtained.*

When adjusting torsions, the default step size is 10 degrees; however, a finer resolution of 1 degree can be achieved by simultaneously holding down the *Shift* key while using the mouse wheel.

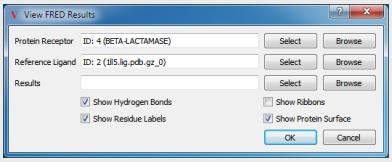
CHAPTER

TEN

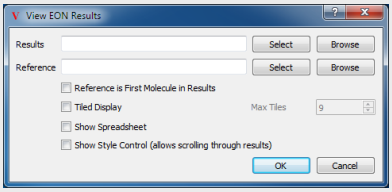
CUSTOM VIEWS

There are a number of very common uses of the 3D display which can take some effort to set up correctly by hand. For this reason, a number of automatically setup custom views are available. These custom views are designed to quickly show in an expected and logical manner the results of various different applications. Support currently exists for viewing FRED, ROCS, or EON results. These views can be accessed by choosing the desired option in the *Tools* menu. An example of the interface to each of these custom views can be seen in *Table: Custom View Inputs*.

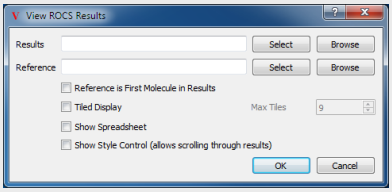
Table 1: Custom View Inputs



FRED Results View



EON Results View



ROCS Results View

The first thing to notice about all the views are the molecule choice buttons. The *FRED View* has three, while the *EON* and *ROCS Views* have only two. Clicking on the *Select* button next to any of these allows selection of a molecule from the currently loaded set. Clicking on the *Browse* button allows selection of a molecule from a file on disk. Following the molecule choice buttons are various check boxes for customizable display options for that view.

10.1 FRED View

The first view is the *FRED View*, the results of which can be seen in *Figure: FRED Results View*. This view sets up the protein's active site (the region around the reference ligand) with a cropped electrostatic surface and the nearby molecular structure. Optionally, it will also display the protein's associated ribbon. Finally, it labels the c-alpha carbons on the protein with residue information.

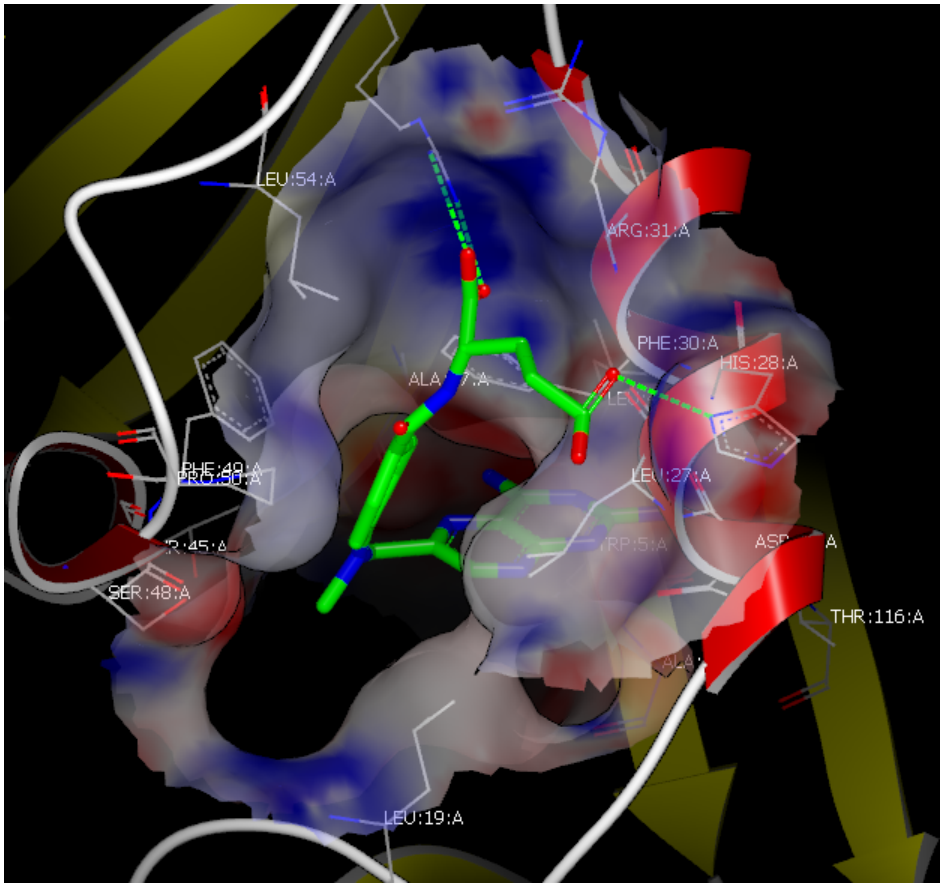


Fig. 1: **FRED Results View**

The *FRED View* then places the reference ligand in the active site and colors the carbons using the "Reference Color" (the default is green). Finally, it loads the optional results and makes the first molecule in that list the *Focused* molecule.

The *FRED View* has several optional parameters, as shown in *Table: Custom View Inputs*. Hydrogen bonds, protein ribbons, residue labels, and the protein surface can all be toggled using the checkboxes at the bottom of the dialog. Furthermore, there are additional preferences with respect to the distance cutoffs used that can be found in the application preferences.

The *FRED View* also adds a custom pane to the Style Control to enable more specific interactions with the scene generated by *FRED View* (see *Figure: FRED Pane*).



Fig. 2: **FRED Pane**

The custom pane contains a row of nine buttons above a "Setup" button which relaunches the original *FRED View* dialog (see *Table: Custom View Inputs*). The first eight buttons toggle the display of hydrogen bonds to the protein, residue labels, ribbons, active site surface, the FRED inner docking contour, the FRED outer docking contour, the

reference ligand, and any other extra molecules that might be associated with the protein such as waters or co-factors. The last button toggles the display of FRED docking constraints (if any) and furthermore contains a drop down menu with a list of all the individual constraints.

Note

Receptors in OEDesignUnit entries are not compatible with *FRED View* at this time

10.2 EON View

The *EON View* has two molecule pickers, one for the result set, and one for the optional reference molecule (see *Table: Custom View Inputs*. Below these are the parameters to the *EON View*. The first checkbox specifies whether or not to use the first result as the reference molecule or not. The second checkbox is the Tiled display option, which sets the application into Tiled mode with the various results displayed in their own individual panes. Next to this checkbox is a numeric entry box to set the max number of tiles in the display. The third check box toggles whether or not to load the spreadsheet display as well. Finally, the last checkbox toggles whether to display the Style Control. This ensures that the *EON* pane in the Style Control is visible, allowing scrolling through the results with that pane's slider.

The *EON View* places the reference molecule in the first tile and colors that molecule using the "Reference Color" (the default is green). The other molecules are displayed in their own tiles. If the molecules contain electrostatic Tanimoto (ET) grids (produced by EON with the -writegrid option), each molecule will have these grid contours displayed for comparison with the reference. An example of this can be seen in *Figure: EON Results View*.

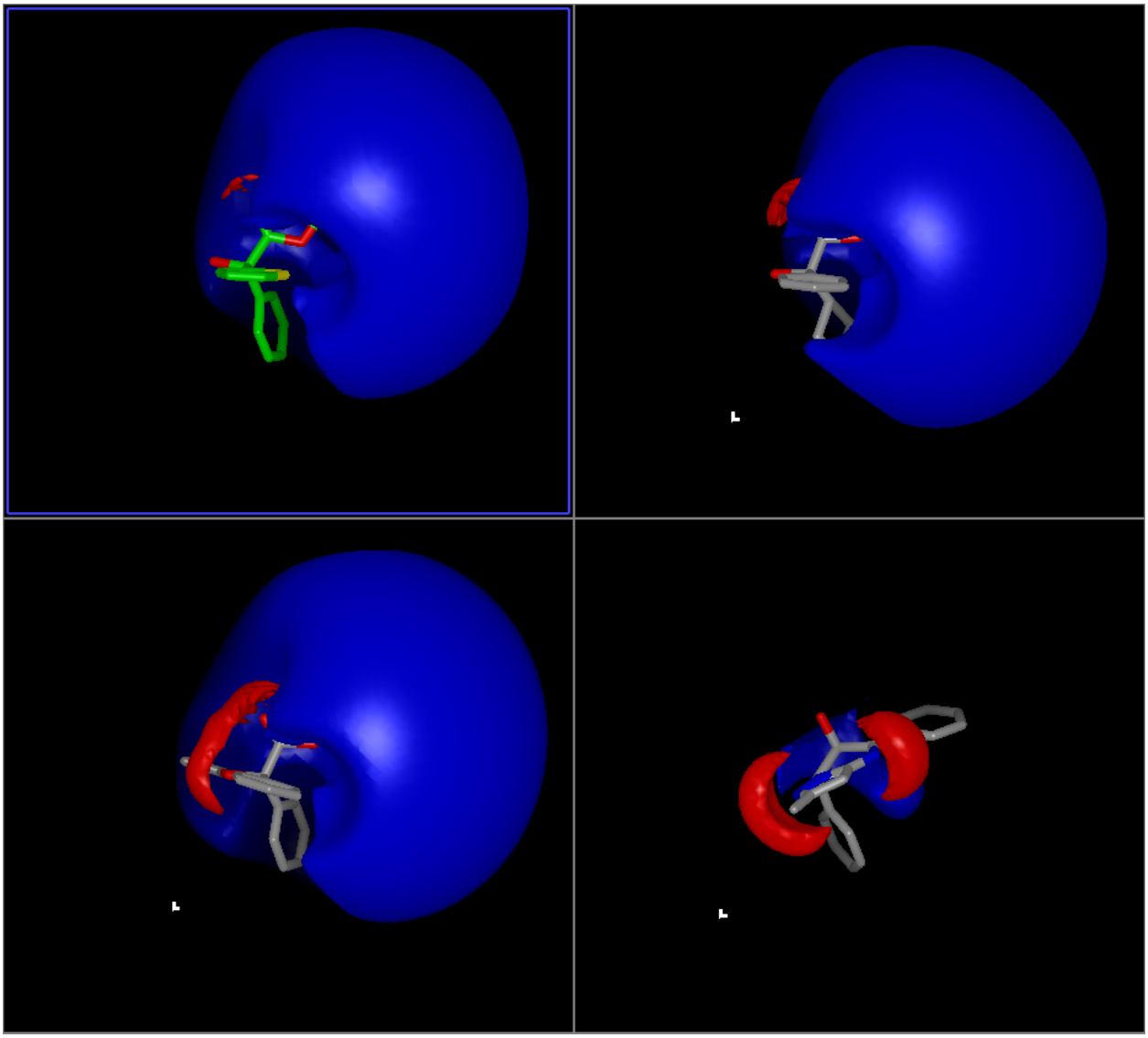


Fig. 3: **EON Results View**

As mentioned previously, the *EON View* adds a custom pane to the Style Control to enable more specific interactions with the scene generated by *EON View* (see *Figure: EON Pane*).



Fig. 4: **EON Pane**

The custom pane contains a row of two buttons and a slider followed by a "Setup" button which relaunches the original *EON View* dialog (see *Table: Custom View Inputs*). The first button toggles the display of the reference molecule. The second button toggles the display of any ET grids that were attached to the molecules by EON. The slider allows scrolling through the results.

10.3 ROCS View

The last custom view is the *ROCS View*. This view, like the *EON View*, has two molecule pickers, one for results and one for a reference molecule (see *Table: Custom View Inputs*). It also has all of the same options as the EON view.

The *ROCS view* colors the reference molecule using the "Reference Color" (the default is green) as well as making that molecule *Locked*. The rest of the results are either overlaid in that same pane or displayed in their own individual pane overlaid on just the reference depending on whether or not Tiled mode was specified. An example of this can be seen in *Table: Custom View Inputs*.

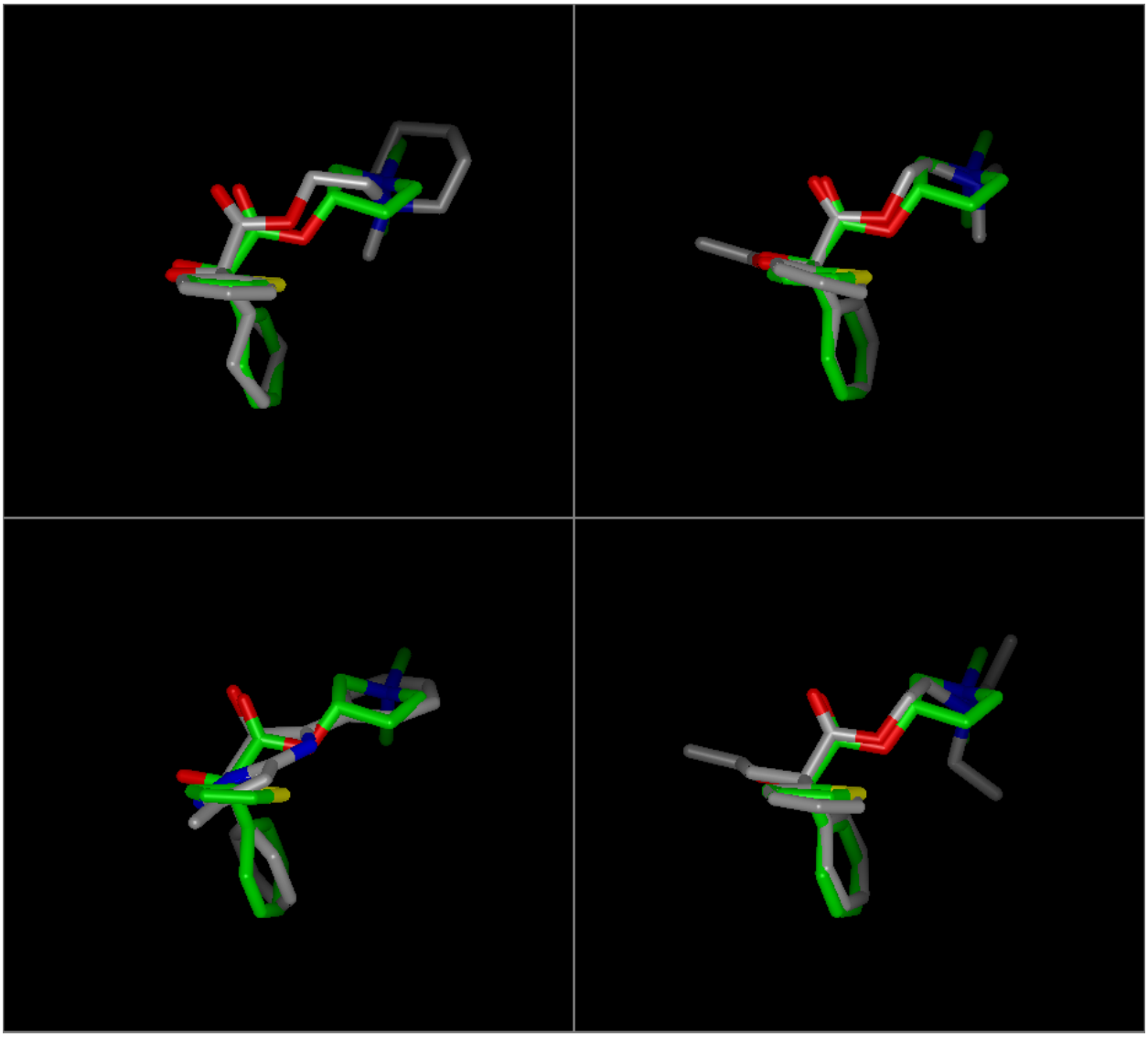


Fig. 5: **ROCS Results View**

As mentioned previously, the *ROCS View* adds a custom pane to the Style Control to enable more specific interactions with the scene generated by *ROCS View* (see *Figure: ROCS Pane*).



Fig. 6: **ROCS Pane**

The custom pane contains a toggle button, a slider, and a "Setup" button which relaunches the original *ROCS View* dialog (see *Table: Custom View Inputs*). The toggle button toggles the display of the reference molecule. The slider allows scrolling through the results.

CHAPTER

ELEVEN

SELECTION LANGUAGE

11.1 Introduction

VIDA has several built-in scripting commands, such as *Visible*, *Select*, *Lock*, *Mark* and *Subset* which take a string argument, where that string is written in the query language described here. This query language is somewhat similar to the command language of the program GRASP. The query language provides a powerful query syntax where atoms, bonds, conformers and molecules may be queried to see if they match certain properties. Triangles, vertices and surfaces may be similarly queried.

A fairly simple example is the following command:

```
Select("ch='A' && rn=10")
```

This command will select all atoms in chain A and in residue 10 for every molecule currently in memory.

A more complex query is the following:

```
Select("id=5 && (r=$hydrophobic || rn=(10,50))")
```

which selects every hydrophobic residue or residues with numbers between 10 and 50, but only matches atoms in the molecule with ID 5.

As the examples indicate, the general syntax of the query language is based on *expressions*. Each expression consists of a *property*, a mathematical *operator* and a *value*. Expressions may be combined using a syntax similar to the C and/or Python programming languages. Boolean operations are supported via the *and* operator `&&`, as well as the *or* operator `||`. Nesting of parts of the query is possible via parentheses `(` and `)`. The operators can also be spelled out as in Python: `and` and `or` are valid for boolean operators.

11.2 Operators

VIDA's query language supports a full range of mathematical operators for testing a property's value against the value(s) specified in the query string. Supported are =, !=, >, <, >=, and <=.

11.3 Lists and Ranges

In addition to being able to specify a single value for a property, it is also possible to specify a *list* of values, or a *range* of values. Lists and ranges are provided as a convenience, since they allow queries matching groups of atoms in a compact manner.

- *LIST* A list is a series of values enclosed in square brackets, for example: `rn=[4,12,38]`. This is exactly equivalent to `(rn=4 || rn=12 || rn=38)`; each listed residue is selected. It is also possible to negate a list: `rn!=[4,12,38]` selects all residue *except* 4, 12, and 38, and is therefore exactly equivalent to `(rn!=4 && rn!=12 && rn!=38)`.

- *RANGE* A range is a pair of values enclosed in parentheses, e.g. `rn=(1,10)`. All residues between 1 and 10, inclusive are selected, so this command is exactly equivalent to `(rn>=1 && rn<=10)`. Negated ranges are also possible: `rn!=(1,10)` is equivalent to `(rn<1 || rn>10)`, and therefore selects atoms *not* in the given range.

Lists and ranges are only valid with the = operator and the != operator. It is not legal (and nonsensical) to try to evaluate an expression such as `rn>(1,10)`.

11.4 Properties

The following properties are defined and may be used for selection:

- *id* - This unsigned integer property limits the match for the part of the query it appears in to match only the object with the given ID as shown in the list window. Example: `id=3`
- *a* - Atom name. This string property matches the atom name. Example: `a=' CA '`. [Note: The spaces before and after the atom name within the quotes are important.] It may also use the pre-defined sets `a=$sch` to match side-chain atoms and `a=$backbone` or `a=$ba` to match backbone atoms.
- *an* - Atom number. This integer property matches the atom number. Example: `an=5`.
- *r* - Residue name. This string property matches the residue name. Example: `r='ALA'`. See also the pre-defined sets in the next section titled "Macros/Pre-defined sets".
- *rn* - Residue number. This integer property matches the residue number. Example: `rn=(1,50)`.
- *ch* - Chain. This string property matches the chain. The chain should be a single letter. Example: `ch='A'`.
- *model* - Model number. This integer property matches atoms with the given PDB-style model number (i.e. for different NMR models). Example: `model=2`.
- *altloc* - Alternate Location indicator. This string property matches atoms with the given PDB-style Alternate-Location property. The string should be a single letter. Example: `altloc='A'`.
- *icode* - Insertion code. This string property matches atoms with the given PDB-style insertion code. Example: `icode='B'`.
- *occ* - Occupancy. This floating point property matches atoms with the given occupancy property. Example: `occ>=0.5`.
- *b* - B-factor. This floating point property matches atoms with the given crystallographic temperature (B) factor. Example: `b>=50.0`.
- *q* - Partial charge. This floating point property matches atoms with the given partial charge. Example: `q>0`.
- *Q* - Formal charge. This floating point property matches atoms with the given formal charge. Example: `Q>0`.
- *radius* - Radius. This floating point property matches atoms with the given radius. Example: `radius<1.4`.
- *ac* - Atom color. This unsigned integer property matches atoms with the given atom color. The color should be specified as a packed integer, e.g. from `GetPackedColor` method on an `OEColor` object. Example: `ac = OEColor(255, 255, 128).GetPackedColor()`.
- *elem* - Element number. This unsigned integer property matches atoms with the given element number. Example: `elem=6`.
- *weight* - Molecular weight. This floating point property matches molecules with the given molecular weight. Example: `weight>10000`.
- *IsAminoAcid* - This boolean property matches atoms whose residue name is the same as an amino acid residue recognized by **OEChem**. Example: `IsAminoAcid=1`. Equivalent to `r=$aa`.
- *IsNucleicAcid* - This boolean property matches atoms whose residue name is the same as a nucleic acid recognized by **OEChem**. A, C, G, T, U. Example: `IsNucleicAcid=1`. Equivalent to `r=$dna`.

- *IsWater* - This boolean property matches atoms which are in a water molecule, as recognized by **OEChem**. Example: `IsWater=1`. Equivalent to `r=$wat`.
- *IsSubstrate* - This boolean property matches atoms which are not protein, nucleic acid or water, as determined by the above queries. Example: `IsSubstrate=1`. Equivalent to `(IsAminoAcid=0 && IsNucleicAcid=0 && IsWater=0)`.
- *type* - This string property may be one of "mol", "atom", "bond", "tri", "vert" or "surf". This limits the matches for the part of the query it appears in to match either molecules, atoms, bonds, triangles, vertices or surfaces, respectively. Example: `type='atom'`
- *index* - This unsigned integer property limits the matches for the part of the query it appears in to items with the given index. Each atom, bond, triangle, etc., has an index assigned when the item is created. Since the indices are generally not exposed to the user, this command is probably of limited utility without OEChem-level access to the molecules in memory. Example: `id=2 && type='atom' && index=10`
- *query* - This string property performs a substructure search where the argument is treated as a SMARTS pattern, and so limits the query to atoms which match the SMARTS pattern. Example: `query='cccn'`
- *subset* - This string property matches all atoms which are in the previously-defined subset with the given name. Subsets may be defined via the `Subset` command, so the `subset` query provides a way of creating shorthand references to other complex queries. Example, first define a subset: `Subset('mysubset','rn=5 && id=2')`. Then the subset may be used as: `subset='mysubset'`.
- *key* - This unsigned integer property limits the match for the part of the query it appears in to match only objects with the given key. Example: `key=100000001`.
- *pkey* - This unsigned integer property limits the match for the part of the query it appears in to match only objects whose parent has the given key. Example: `pkey=100000001`.

11.5 Macros/Pre-defined sets

In addition to explicitly naming residues and atoms, VIDA defines several macros which may be used in query strings. These macros are prefixed with a dollar sign ($). Macros are generally used with the "residue name" (i.e. `r=`) property, although a few are used with the "atom name" (i.e. `a=`) property. The definitions for these sets are largely borrowed from RasMol.

- `r=$aliphatic`: ALA, GLY, LEU, VAL, ILE, PRO
- `r=$hydroxyl`: SER, THR, TYR
- `r=$sulfur`: CYS, MET
- `r=$aromatic`: TYR, HIS, TRP, PHE
- `r=$charged`: ASP, GLU, ARG, LYS
- `r=$amide`: GLN, ASN
- `r=$hydrophobic`: ALA, GLY, LEU, VAL, ILE, PRO, MET, PHE, TRP
- `r=$polar`: SER, THR, CYS, TYR, HIS, ASP, GLU, ASN, GLN, ARG, LYS
- `r=$neutral`: ALA, GLY, LEU, VAL, ILE, PRO, SER, THR, CYS, MET, PHE, TYR, TRP, ASN, GLN
- `r=$acidic`: ASP, GLU
- `r=$basic`: ARG, LYS
- `r=$small`: ALA, GLY, SER
- `r=$medium`: VAL, PRO, THR, CYS, ASP, ASN
- `r=$large`: ILE, MET, PHE, TYR, HIS, TRP, GLU, GLN, ARG, LYS

- r=$cyclic: HIS, PRO, TYR, TRP, PHE
- r=$dna: ADE, A, GUA, G, CYT, C, THY, T, URA, U
- r=$aa: all amino acids
- r=$at: ADE, A, THY, T, URA, U
- r=$cg: CYT, C, GUA, G
- r=$purine: ADE, A, GUA, G
- r=$pyrimidine: CYT, C, THY, T, URA, U
- r=$wat: WAT, H2O, HOH, TIP, SOL
- r=$substrate: this is defined as r!=$wat && r!=$aa && r!=$dna.

There are two atom name macros as well, a=$backbone (which may be abbreviated as a=$ba), which matches specifically protein backbone atoms, and a sidechain macro, a=$sch, which matches protein sidechain atoms. Both of these can be negated (a!=$backbone or a!=$sch).

11.6 Scripting with ScratchScope

In addition to the built-in commands Visible, Select, Lock, Mark and Subset, it's quite straightforward to use a selection string with any command which operates on a *scope*, by binding the selection string to the *ScratchScope*.

For example the following function defines an atom coloring command ac:

```
def ac(color,str):
  Subset("scratch",str)  # use string to make named subset
  ScratchSet("scratch")  # bind it to scratch scope
  AtomColorSetScoped(OEggColor(color),ScratchScope) # use it
```

With two small helper functions, these types of functions can be made even easier to create:

```
def ScratchSubset(str):
    Subset("scratch",str)
    ScratchSet("scratch")
    return ScratchScope

def ScratchSubsetList(str):
    Subset("scratch",str)
    ScratchSet("scratch")
    return ScratchList()
```

With these functions defined, the ac function above and functions similar to it can be easily defined. For example:

```
def ac(color,str):
    AtomColorSetScoped(color,ScratchSubset(str))

def lc(color,str):
    LabelColorSetScoped(color,ScratchSubset(str))

def center(str):
    ViewerCenterSetScoped(ScratchSubset(str))

def show(str):
```

(continues on next page)

(continued from previous page)

```
    Visible(ScratchSubsetList(str), True)

def hide(str):
    Visible(ScratchSubsetList(str), False)
```

In this manner, any of the VIDA scripting commands which take a scope argument can be easily expanded to take a selection string.

CHAPTER

TWELVE

SCRIPTING

VIDA is a scriptable application which contains a built-in Python interpreter. Python is an interactive, interpreter, strongly and dynamically typed language. More information about Python can be obtained from the Python website. The website contains many useful tutorials and other related information.

All user interaction with the application goes through a Python layer even if the user is not aware of it. For instance, every button in VIDA is associated with a Python command that is sent to the internal interpreter when that button is pressed. The same integration is true of all the menu items and other GUI controls in VIDA.

12.1 Journal File

The journal file is a recording of every action that is performed within a given session of VIDA. In fact, the journal file is itself a Python script which can be run to regenerate the state of the application at the time the file was written.

The journal file is written to `journal.py` in the user directory (see *User Directory*). In the unlikely event that VIDA were to quit unexpectedly, the journal file corresponding to that run will be saved in the same directory but with a different name containing a time stamp associated with when the application was started (e.g. `journal_WedMar181507132009.py`). Journal files are particularly useful for debugging unexpected application behavior and are of extreme benefit to the application developers when tracking user-reported bugs. The inclusion of journal files (if possible) with bug reports is greatly encouraged.

12.2 Startup File

The startup file is a Python script located in the user directory (see *User Directory*) named `startup.py`. This script (if present) is executed each time VIDA starts.

The startup file provides the user with a convenient mechanism to customize the appearance and functionality of VIDA using Python.

12.3 Scripts Directory

In addition to interpreting the startup file (see *Startup File*) at runtime, VIDA also executes any Python scripts contained in the `scripts` directory of the user directory (see *User Directory*). This capability provides a simple mechanism to quickly and easily add or remove custom functionality to VIDA without having to change the startup file.

12.4 Scripting API

The complete details of the VIDA Python interface and API are fully described in the appendix.

12.5 Scripting Window

The scripting window provides an interactive user interface to the embedded Python interpreter. It also displays the output from any Python commands issued to VIDA as well as any warnings or errors reported by the application.

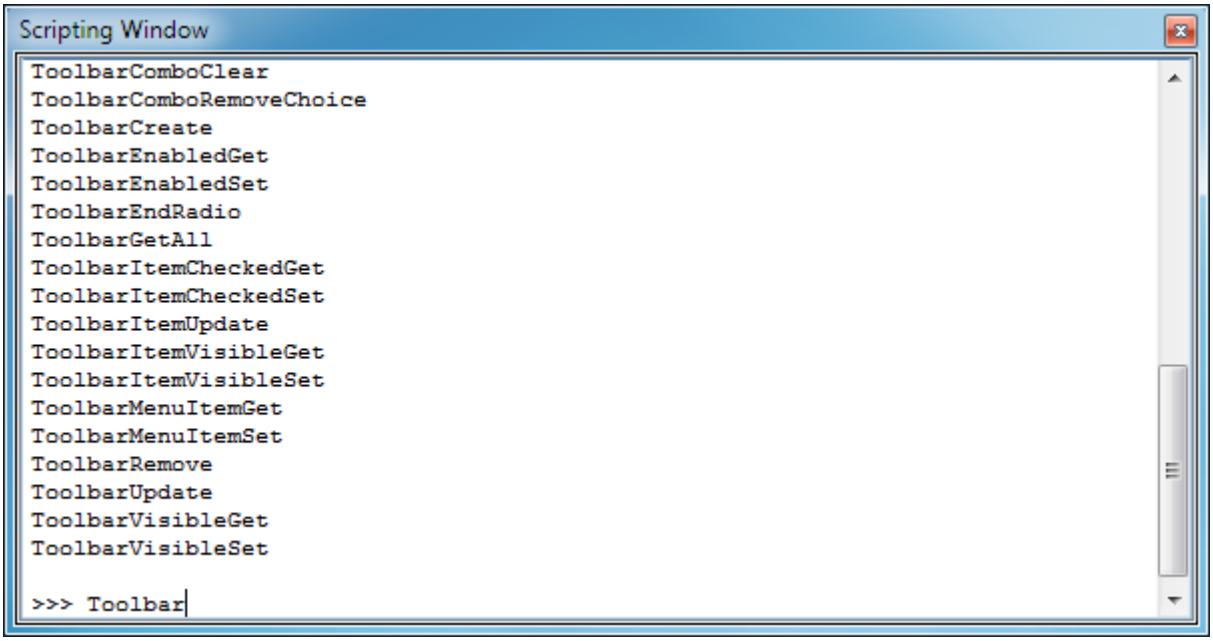


Fig. 1: **Scripting Window**

The scripting window supports tab completion of the built-in API methods and hitting the tab key again after tab completion will display the API documentation associated with the command directly in the window.

The window history can be cleared by typing the following command `InterpreterClear()` into the scripting window.

12.6 OpenEye Toolkits

VIDA provides the ability to import any of the OpenEye Python toolkits directly into the interpreter. Once a toolkit has been imported, its entire functionality is available for use within VIDA. A valid license for the imported toolkit is required in order for the import to succeed. An exception will be thrown at import time if a valid license cannot be found. All of the toolkits come directly bundled with the application, so there is no additional need to perform any installation.

Molecule, grid, and surface objects can be passed back and forth between the main application and the Python interpreter using a handful of methods. For stability reasons, copies of these objects must be "checked out" of VIDA, where they can be accessed by the toolkits. If any modifications are made to the objects in the interpreter, these modifications can be applied to the associated objects in VIDA by "checking in" the modified object. One exception to this rule, is that it is possible to make display changes to the original objects by calling the built-in API functions and passing in the unique key obtained from the object using the `GetKey()` method. A number of examples follow.

12.6.1 OEChem Objects

Most of the standard scripting functions in VIDA that operate on atoms, bonds, or molecules take a unique key as a parameter. VIDA adds a `GetKey()` function to every **OEChem** object (e.g., OEMol, OESurface, OEAtomBase, etc.) that is accessed from VIDA. In addition, VIDA adds the following methods to all accessed objects for convenience:

- `IsActive()`
- `IsLocked()`
- `IsMarked()`
- `IsSelected()`
- `IsVisible()`
- `SetLocked( value = True )`
- `SetMarked( value = True )`
- `SetSelected( value = True )`
- `SetVisible( value = True )`

Access to a given molecule is provided through the `MoleculeGet` function. The molecule object returned is a copy of the actual molecule in use in VIDA. However, this copy can still be used to retrieve information about that original molecule as well as to alter the original molecule's display and associated data through the standard VIDA API.

12.7 Customizing the User Interface

The standard VIDA scripting API provides a wide variety of options to customize both the menus and toolbars and is detailed extensively in the appendix.

More advanced customization and extension of the user interface is possible through the PySide2 module, which provides Python wrappers for the Qt widget library used by VIDA for its user interface. Using this module within VIDA enables the user to create and interact with entirely new user interface elements. The specific details of the PySide2 module are described in the section below. For more details on Qt, please visit these links: Qt Company, The Qt Project and Qt for Python project.

12.7.1 PySide2

PySide2 is developed by the Qt for Python project and is supported by the Qt Company.

PySide2 is a complete wrapper around the Qt library which enables complex user interface customization and development.

In addition, to a complete and fully functional API for building widgets, PySide2 can also build widgets using the output from Qt Designer. Qt Designer is a graphical tool for designing and building user interface elements from Qt components. Designer can save the finished widget design to an XML file (.ui) which can then be passed to a widget object for creation in Python. The user must have a valid Qt license in order to use Qt Designer.

For more details on PySide2, please visit: Qt for Python project

CHAPTER

THIRTEEN

EXTENSIONS

VIDA provides an *Extension* facility to allow users to modify and extend the user interface and general functionality of the program using Python scripting. VIDA extensions can expose functionality of all the OpenEye Toolkits (with appropriate licenses) or third party applications, and can also be used to customize VIDA's user experience. They provide a convenient way to carry out regular tasks, streamline corporate workflows, and connect VIDA to other systems. A simple example might be the ability to search a corporate database by corporate compound ID and load that molecule into VIDA.

An individual extension is implemented as a Python module which, once installed, is imported into VIDA at startup. Extensions can be added, removed, updated, and/or disabled from the *Extension Manager*.

13.1 Example Extensions

A number of example extensions are available in the VIDA distribution for use as-is or as starting points for writing custom extensions (Python source code for these is included). The provided extensions are:

- **Alternate Location Browser**: Browse through alternate conformations of a protein (as defined in a PDB file) and take snapshots of selected configurations. *(Additional license required: none)*
- **Binding Site Align**: Align proteins based on binding site residues. *(Additional license required: none)*
- **Bump Checker**: Displays intermolecular bump interactions. *(Additional license required: none)*
- **Center on Ligand**: Automatically locates and centers the view on a ligand in a protein-ligand complex. *(Additional license required: none)*
- **Color by Atom Properties**: Colors atoms by atom-level properties. *(Additional licenses required: none)*
- **Color by SD Data**: Colors molecules by SD data values. *(Additional licenses required: none)*
- **Docking**: Performs docking of ligands against pre-prepared receptors. *(Additional license required: **OEDocking TK*)*
- **FastROCS Client**: Submits a search to a FastROCS server, and loads the results into VIDA. *(Additional license required: none. Also requires access to a FastROCS server.)*
- **Fit**: Performs shape-based alignments. *(Additional licenses required: **Shape TK*, **Omega TK**)*
- **Interaction Map**: Displays pattern-based intermolecular interactions. *(Additional licenses required: none)*
- **Ligand Minimizer**: Perform minimizations in an active site. *(Additional licenses required: **Szybki TK*)*
- **Molecule Alignment**: Perform molecular alignments based on shape & color, sequence or RMSD. *(Additional licenses required: none. Optional: **Shape TK*)*
- **OMEGA**: Generate molecule conformations using the OMEGA software. *(Additional licenses required: **Omega TK*)*

- **Void Volumes**: Create void volumes around a ligand in a protein-ligand complex. *(Additional licenses required: **Spicoli TK*)*
- **3D Molecule Filter**: Filters lists of aligned molecules, *e.g.*, ROCS results, based on positions of features in 3D. *(Additional licenses required: none)*

Although these extensions are distributed with VIDA, you will need to install them with the *Extension Manager* before using them.

13.2 Extension Manager

The *Extension Manager* is accessed under *Edit* > *Extensions*, and provides a way to add, activate, and remove extensions. Its use is illustrated here for the example extensions that ship with VIDA.

13.2.1 Add Extensions

To "subscribe" to the set of example extensions, open the *Extension Manager*, and click on the *Manage Subscriptions...* button to open the Manage Subscriptions dialog, then click the folder icon to browse to the directory where the example extensions are located. This will be under the OpenEye installation directory, in the subdirectory examples/vida/[version]/extensions.

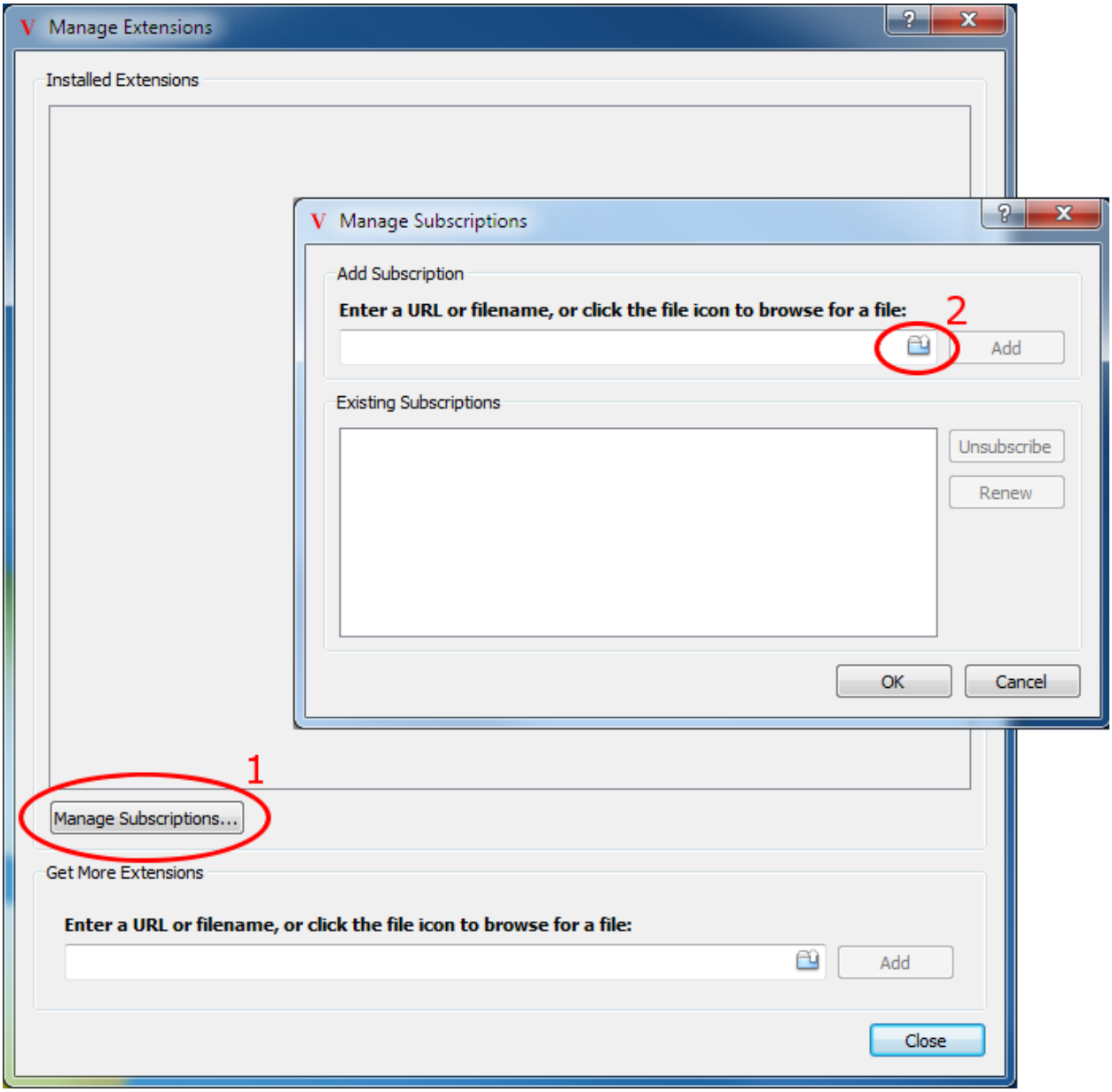


Fig. 1: **Basic steps to add extensions**

There is a file in this directory named examples.pyxs. This type of file is called a subscription, and simply contains a list of URLs pointing to available extensions. Select this file, then click the Open button to install all of the subscription's extensions. The list of extensions in the subscription will be shown in the *Manage Subscriptions* dialog. You can disable individual extensions using the checkboxes that appear beside each extension name.

Once a subscription has been added, an entry for each available extension will be displayed. Every extension has a name and a version (to enable updates), as well as a brief description and a list of any other required OpenEye toolkits for the extension to run. Additionally, each extension appears with four buttons: **Info**, **Refresh**, **Delete**, and **Enable**.

All of the example extensions, once added, will appear in the *Extensions* menu in VIDA's main window, although custom extensions can be made to show up in other menus, as buttons in toolbars, or as other user interface elements.

13.2.2 Installed Extensions

The *Installed Extensions* list can be seen in *Figure: Installed Extensions*. It provides the user with the ability to view which extensions are installed as well as to update, uninstall, or disable the extension.

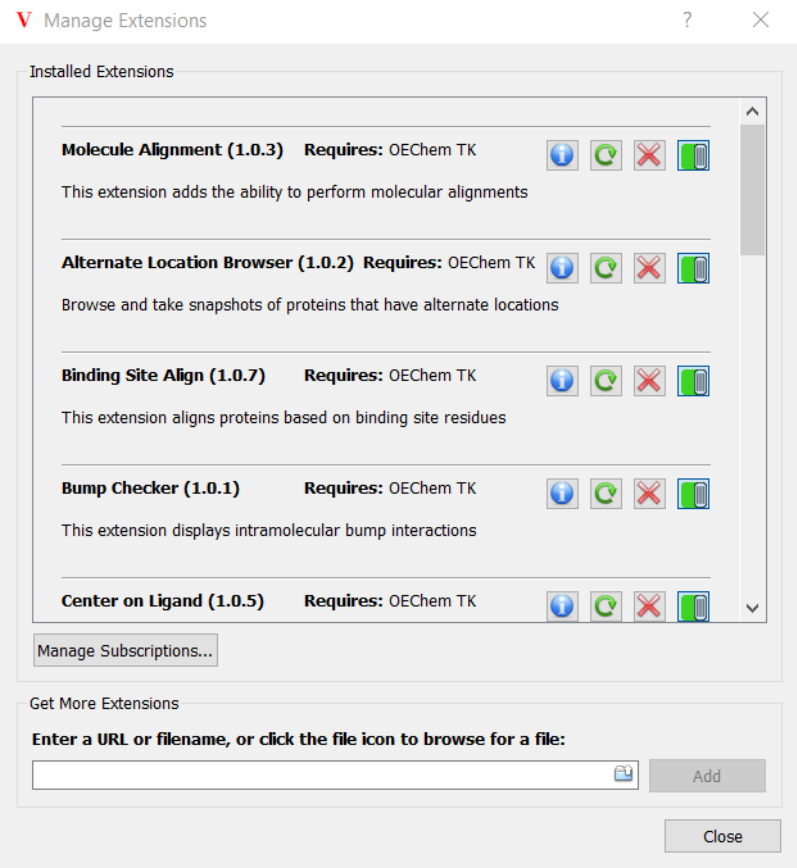


Fig. 2: **Installed Extensions**

Clicking on the **About** button for an extension will display more detailed information about the extension than is seen in this section. Clicking on the **Update** button will cause VIDA to check for an updated version of the extension and if found, install it. Clicking on the **Delete** button will cause the extension to be uninstalled when VIDA exits. Clicking on the **Disable** button will prevent this extension from being imported the next time VIDA is run. A disabled extension will still be listed in the *Installed Extensions* section and can be reenabled by clicking on the **Enable** button.

13.2.3 Updating Extensions

VIDA checks for extension updates at startup and will install the updates directly if found. Extensions can also be manually updated using the **Update** buttons in the Installed Extensions list.

13.2.4 Managing Subscriptions

The **Manage Subscriptions...** button opens a dialog with a list of your current subscriptions. In this dialog you can add or remove a subscription. Removing a subscription will uninstall all of its extensions. You can also renew a subscription, which will update all of its extensions at once. In the **Existing Subscriptions** list, you can enable or disable individual extensions using the checkboxes next to the extension names.

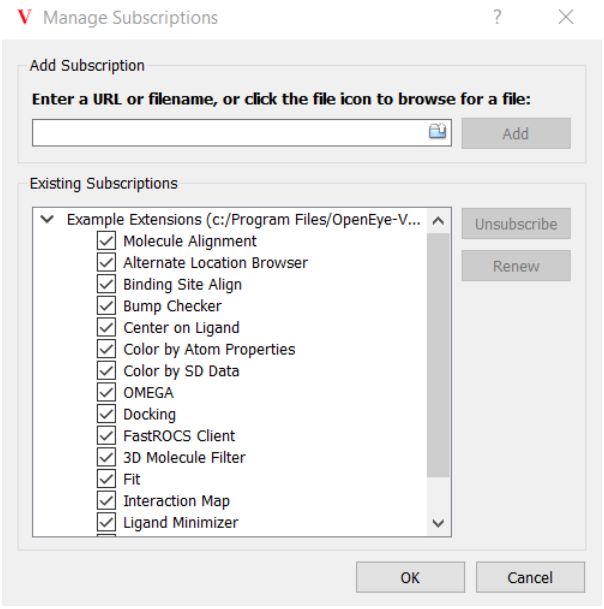


Fig. 3: **The Manage Subscriptions Dialog**

13.3 Creating Extensions

Creating an extension for VIDA is very easy. At its simplest level, a VIDA extension is simply a directory containing an `__init__.py` file and any associated data files or additional scripts. Once this directory is installed, the Python interpreter will import this extension every time the application starts. The process of importing the extension will cause all the code in the `__init__.py` file to be run.

13.3.1 The OEVidaExtension Base Class

Using the OEVidaExtension base class can simplify the creation of an extension script. When an extension is derived from OEVidaExtension, the following are automatically handled for you:

- Importing of the PySide2 and other modules
- Adding the extension to the Extensions menu
- Loading of a Qt .ui file and creating a docked window from it
- Providing Python access to the individual widgets in a Qt .ui file
- Calling a one-time initialization function within the extension script

The basic steps for creating an extension using OEVidaExtension are:

1. If your extension needs a user interface, create a .ui file using Qt Designer. Give it the same first name as your extension.
2. Create a .pyx file (*see below*) containing information about the extension. The "title" record in this file is the part OEVidaExtension cares about. This step is optional during development of an extensions, but must be done before the extension is installed or deployed.
3. Create a Python script with a class derived from OEVidaExtension
4. Give your class an InitializeUI() method, where initialization of the user interface takes place
5. Create an object from your class. You're done!

The following script illustrates basic use of the OEVidaExtension class:

13.3.2 Details for Using the OEVidaExtension Class

The constructor for OEVidaExtension accepts the following arguments:

- **extensionPath** *(Required)*: This should be the `__file__` special variable, and is used to find the .pyx and .ui files.
- **menuName** *(Optional)*: Specifies what menu the extension should appear in. If omitted, this defaults to 'Extension'. If you want to specify a submenu, *e.g.*, 'Extensions > Proteins', use the form '/Extensions/Proteins'. Note the leading forward slash character.
- **windowSubmenu** *(Optional)*: This specifies an optional name of a submenu in the Window menu, where this extension's window can be toggled on and off. The default value is '/Extensions', which puts an entry in an Extensions submenu of the Window menu.
- **command** *(Optional)*: Specifies a Python command to be issued when this extension is invoked from the menu. If left blank, selecting the extension from the menu will make its window visible.
- **uiFile** *(Optional)*: If the .ui file's name does not match the extension's name, it can be specified here.
- **title** *(Optional)*: Specifies a title to be used in the menus and on the title bar of the extension's window. Used only if a title cannot be found in a .pyx file. This can be useful while developing an extension, if a .pyx file has not yet been created.

The constructor will create two member variables that may be useful:

- **self.ui**: This is an object whose members represent the named widgets within the extension's .ui file. For example, if a QPushButton named "myPushButton" is defined in the .ui file, you can refer to it using `self.ui.myPushButton` within the script.
- **self.dockWidget**: This is a QDockWidget containing the extension window. If your script needs to hide, show, or otherwise access the extension's window, it can do so using this variable.

A class derived from OEVidaExtension must implement a method called InitializeUI(), which performs one-time initialization of the class. This method is also responsible for connecting the Qt widgets in the user interface to the methods in your class.

Some extensions won't need a .ui file (for example, a menu item that performs an action directly). In such a case, be sure to provide a `command` argument to the constructor, since the default menu action is to display the extension's (in this case nonexistent) window.

13.3.3 Extension Preferences

The OEVidaExtension class provides several methods for managing preferences for extensions. This allows extensions to remember values from one VIDA session to the next. These preferences are stored as key-value pairs, and are preserved in state (.oes) files so that extensions can be set to a particular starting point when a state file is loaded. Preferences can be private to an individual extension, or can be "shared" among all extensions. The methods for managing preferences are:

- `SetPref(self, key, value, shared = False)` – Sets the named preference to the specified value. In this and other methods, the *shared* parameter specifies whether this preference is to be shared by all extensions. If *shared* is true, other extensions will be notified when its value has changed, via the SharedPrefChanged method (see below).
- `GetPref(self, key, defaultValue, shared = None)` – Returns the value of a preference. If no such preference exists yet, *defaultValue* is returned. If *shared* is True, a shared preference value is returned. If *shared* is False, a private preference value is returned. When *shared* is None, the private value is returned if it exists; otherwise the shared value is returned.

- `RemovePref(self, key, shared = False)` – Removes a preference.
- `ClearPrefs(self, shared = False)` – Removes all preferences for an extension.
- `SharedPrefChanged(self, key, value)` – Called when a shared preference has changed. The default implementation of this function does nothing. You can reimplement it in a class derived from OEVidaExtension if you want your extension to do something when shared preferences change.

13.3.4 Notification Methods

The OEVidaExtension class provides two methods to allow extensions to be notified when VIDA is finished starting and when it is about to close. The default implementations of these methods do nothing, but you can reimplement them in derived classes to perform actions on startup and shutdown. These methods are:

- `StateReset(self)` – Called when the state of VIDA has been reset. This happens at program startup, when a state (.oes) file has been loaded, or when all objects have been cleared using the `DeleteAll()` function.
- `AppShutdown(self)` – Called when VIDA is about to exit, but before preferences have been saved. Reimplement this method if your extension needs to save some preferences or perform other tasks before closing down.

13.4 Distributing Extensions

Extensions are distributed as directory archives (e.g. tar or zip files) coupled with a meta-file containing detailed information about the extension. The meta-file is a simple plain text file named with a .pyx file extension. This file is intended to provide all the associated details about the extension in an easily accessible way. The data contained in the file includes:

- Author
- Title
- Description
- Version
- Name of the directory to be extracted
- Required version of VIDA
- Any required OpenEye toolkits
- File's own URL (so the file always knows where it came from)
- URL of the directory archive containing the extension. Note that this URL can be a "short" filename, which implies that the archive can be found in the same directory as the .pyx file.
- An optional specification of the order in which the extension should be loaded

When distributing multiple independent extensions it may be useful to organize them together into groups called *Extension Sources* or subscriptions. VIDA may subscribe to an *Extension Source* to get a list of available extensions for installation. The *Extension Source* itself (as described in *Add Extensions*) is a simple plain text file named with a .pyxs file extension which contains a title followed by a list of URLs to a collection of extension meta-files. These URLs can be simple file names, in which case VIDA will assume these reside in the same location as the .pyxs file.

13.4.1 Automatic Subscriptions

Users can be automatically subscribed to an extension source through the environment variable VIDA_EXTENSIONS_URL. When this environment variable is set to the URL of a .pyxs file, VIDA will install the full set of contained extensions without requiring any user intervention. To automatically subscribe to multiple extension sources, set the VIDA_EXTENSIONS_URL variable to a comma-separated list of URLs.

CHAPTER

FOURTEEN

PREFERENCES

Every user has his or her own individual preferences with regards to how molecules, grids, and surfaces should look and how applications should behave. For this reason, a Preferences dialog is available which allows customization of the application to the user's preference. An snapshot of the Preferences dialog can be seen in *Figure: Preferences*.

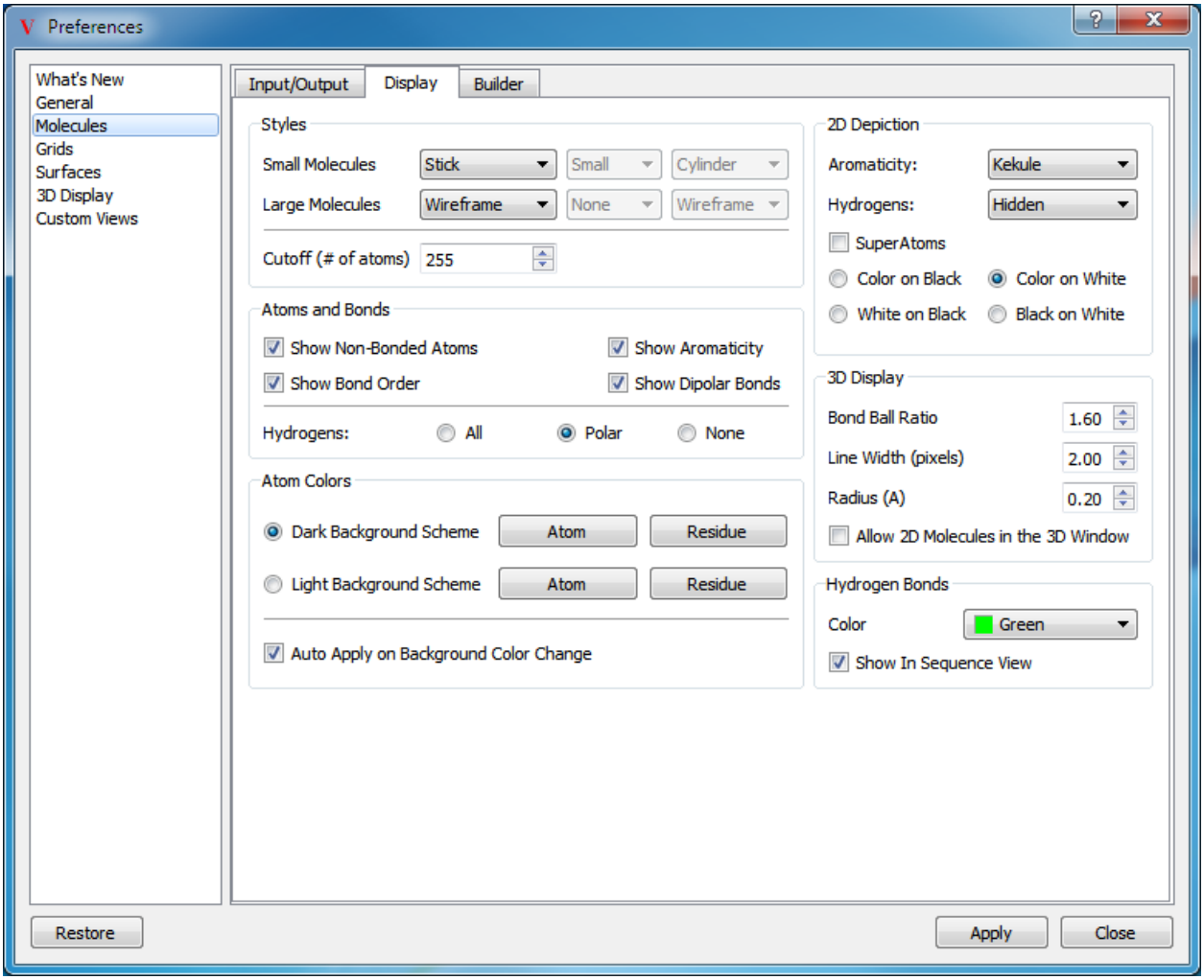


Fig. 1: **Preferences**

On the left-hand side of the dialog is column containing preference categories. These categories include: *General*,

Molecules, *Grids*, *Surfaces*, *3D Display*, and *Custom Views*. Clicking on any of these categories will update the right-hand side of the widget to display the options corresponding to the selected category.

At the bottom of the dialog, the *Restore* button will restore the current preferences to the default ones. The *Apply* button will apply and save the current preferences.

Preferences are stored in a binary file (`preferences.oeb`) in the *User Directory*. While the preference file shares the same file extension as the OpenEye's binary database file, it cannot be read into VIDA using the "Open" menu item in the *File* menu. The preferences file is loaded automatically when the applications starts and is saved back to disk when the application exits. Deleting this file is equivalent to clicking on the *Restore* button in the dialog box.

There is also a file in this same directory which called VIDA.ini which contains machine specific settings like the list of recent files, preferred layouts, and hardware stereo options. Deleting this file will restore these settings to the defaults as well.

This directory can be opened from within VIDA by selecting the "Open User Directory" menu item in the *Help* menu.

CHAPTER

FIFTEEN

MENUS

This chapter contains a listing of the menu options available in VIDA. Where possible, each section also contains a listing of the Python code that is run when that option is selected. All Python menu commands are wrapped in a standard exception handling block:

```
try: exec(<command>)
except Exception, e:
    if 'Canceled' not in str(e):
        DoMenuError(e)
        raise;
```

15.1 File

15.1.1 Open

```
Open(PromptFilenames([], 'molecule,grid,surface,query,python,state'))
```

15.1.2 Open State

```
Open(PromptFilename('','Open','state'))
```

15.1.3 Open Special

From PDB...

```
LoadMoleculesFromPDB(PromptString('Enter PDB code(s)'))
```

Script...

```
Open(PromptFilename('', 'Open', 'python')
```

MTZ...

```
DoOpenMTZ()
```

15.1.4 Clear

Marked

```
DeleteScoped(MarkedScope, True)
```

Not Marked

```
DeleteScoped(MarkedScope, False)
```

Visible

```
DeleteScoped(VisibleScope, True)
```

Not Visible

```
DeleteScoped(VisibleScope, False)
```

15.1.5 Clear All

```
DeleteAll()
```

15.1.6 Save

```
try:
    _tmp_keys = PromptKeys()
    if _tmp_keys:
        _tmp_sname = PromptSaveFilename(_tmp_keys)
        if _tmp_sname:
            Save(_tmp_sname, _tmp_keys)
except:
    pass
```

15.1.7 Save Marked

```
try:
    _tmp_keys  = GetMoleculesScoped(MarkedScope)
    _tmp_keys += GetGridsScoped(MarkedScope)
    _tmp_keys += GetSurfacesScoped(MarkedScope)
    if _tmp_keys:
        _tmp_sname = PromptSaveFilename(_tmp_keys)
        if _tmp_sname:
            Save(_tmp_sname, _tmp_keys)
except:
    pass
```

15.1.8 Save State

```
try:
      _tmp_filters = 'OpenEye State Version 2 ( *.oes ) ;;'      \
                                        'OpenEye Mini State Version 2 ( *.oes ) ;;' \
                                        'OpenEye State Version 1 ( *.oes ) '
      _tmp_results = PromptSaveFilename(_tmp_filters)
      if len(_tmp_results) == 2:
          SaveStateFilter(_tmp_results[0], _tmp_results[1])
except:
    pass
```

15.1.9 New List

From AND

```
try:
    _tmp_ids = PromptIDs(['Lists'], 0)
    if _tmp_ids:
        ListNewAnd(_tmp_ids)
except:
    pass
```

From OR

```
try:
    _tmp_ids = PromptIDs(['Lists'], 0)
    if _tmp_ids:
        ListNewOr(_tmp_ids)
except:
    pass
```

From XOR

```
try:
    _tmp_ids = PromptIDs(['Lists'], 0)
    if _tmp_ids:
        ListNewXor(_tmp_ids)
except:
    pass
```

From Marked

```
ListNewMarked()
```

15.1.10 New Molecule

From Name

```
try:
    _tmp_smiles = PromptSmiles('New Molecule')
    if _tmp_smiles:
        _tmp_mol = MoleculeNew(_tmp_smiles)
```

(continues on next page)

(continued from previous page)

```
            if _tmp_mol:
                MoleculeGenerateCoords(_tmp_mol, 1)
                Active(_tmp_mol)
    except:
        pass
```

From Merge

All three options under this menu (*Scoped*, *Marked*, and *Visible*) execute the following code.

```
    try:
        _tmp_name = PromptString('Name:')
        if _tmp_name:
            _tmp_id = MoleculeMergeScoped(SCOPE, _tmp_name)
            if _tmp_id:
                Active(_tmp_id)
    except:
        pass
```

The SCOPE parameter is assigned a value according to which option was selected (SelectedScope, MarkedScope, and VisibleScope respectively).

From Split

Components

```
    ids = MoleculeNewSubsetScoped(0,ActiveThenVisibleScope,1)
    if ids:
        Active(ids[0])
```

Selected

```
    ids = MoleculeNewSubsetScoped(0,ActiveThenVisibleScope,SelectedScope,1)
    if ids:
        Active(ids[0])
```

Marked

```
    ids = MoleculeNewSubsetScoped(0,ActiveThenVisibleScope,MarkedScope,1)
    if ids:
        Active(ids[0])
```

15.1.11 Import

Spreadsheet

```
    _tmp_fname = ''
    try:
        _tmp_fname = PromptFilename('', 'Open', 'data')
        if _tmp_fname:
            eval( SpreadsheetPromptImport(_tmp_fname) or 'None' )
    except:
        pass
```

15.1.12 Export

Script History

```
try:
    _tmp_fname = PromptFilename('journal.py', 'Save', 'python')
    if _tmp_fname:
        AppScriptSave(_tmp_fname)
except:
    pass
```

Screenshot

```
MainWindowScreenshotPrompt()
```

Spreadsheet

```
try:
    _tmp_fname = PromptFilename('', 'Save', 'data')
    if _tmp_fname:
        eval( SpreadsheetPromptExport(_tmp_fname) or 'None' )
except:
    pass
```

POV-Ray

```
try:
    _tmp_fname = PromptFilename('', 'Save', '')
    if _tmp_fname:
        ViewerExportPOVRAY(_tmp_fname)
except:
    pass
```

15.1.13 Recents

All of the options in this menu call the following command:

```
OpenWithRecents(FILENAME)
```

The FILENAME parameter is assigned to the filename specified in the option that was selected under this menu.

15.1.14 Recent Directories

All of the options in this menu call the following command:

```
PromptOpenWithRecents('molecule,grid,surface,query,python,state', FILENAME)
```

The FILENAME parameter is assigned to the filename specified in the option that was selected under this menu.

15.1.15 Exit

```
Quit()
```

15.2 Edit

15.2.1 Undo

```
Undo()
```

15.2.2 Undo History

15.2.3 Redo

```
Redo()
```

15.2.4 Redo History

15.2.5 Copy

```
Copy()
```

15.2.6 Copy Image

```
MainWindowScreenshot()
```

15.2.7 Paste

```
PasteMolecules()
```

15.2.8 Select

All Visible

```
SelectAllMolecules()
```

Expression

```
try:
    _tmp_selexpr = PromptString('Selection Expression')
    if _tmp_selexpr:
        Select(_tmp_selexpr)
except:
    pass
```

Query

```
try:
    _tmp_query = PromptQuery()
    if _tmp_query:
        SelectByQuery(_tmp_query, False, VisibleScope)
except:
    pass
```

Invert

```
SelectInvert()
```

None

```
ClearSelection()
```

15.2.9 Mark

All Visible

```
MarkObjectsByScope(VisibleScope)
```

Query

```
try:
    _tmp_query = PromptQuery()
    if _tmp_query:
        SelectByQuery(_tmp_query, True)
except:
    pass
```

None

```
ClearMarked()
```

15.2.10 Hide

All

```
ClearVisible()
```

Selected

```
HideScoped(SelectedScope, True)
```

Marked

```
HideScoped(MarkedScope, True)
```

Not Selected

```
HideScoped(SelectedScope, False)
```

Not Marked

```
HideScoped(MarkedScope, False)
```

None

```
HideNoneScoped()
```

15.2.11 Find

By LINGO Similarity

Creates a new list of molecules, based on the LINGO similarity measure to a query molecule. LINGO is a fast text-based similarity measure that uses the canonical, isomeric SMILES strings of molecules for comparisons.

See also

- [Vidal-2005]

```
try:
    _tmp_key    = PromptKey()
    _tmp_cutoff = PromptFloat('Similarity Cutoff', 0.4)
    if _tmp_key.IsValid() and _tmp_cutoff > 0.0:
        ListNew('Find Results', FindBySimilarityScoped(_tmp_key, _tmp_cutoff))
except:
    pass
```

By SMARTS

```
try:
    _tmp_query = PromptQuery()
    if _tmp_query:
        ListNew('Find Results', FindBySMARTSScoped(_tmp_query))
except:
    pass
```

By Query Molecule

```
try:
    _tmp_key = PromptKey()
    if _tmp_key.IsValid():
        ListNew('Find Results', FindByQueryScoped(_tmp_key))
except:
    pass
```

By Title

```
try:
    _tmp_title = PromptString('Title Search')
    if _tmp_title:
        ListNew('Find Results', FindByTitleScoped(_tmp_title))
except:
    pass
```

15.2.12 Find on Disk

```
FindOnDisk()
```

15.2.13 Clear Scripting Log

```
InterpreterClear()
```

15.2.14 Extensions

```
ExtensionsEdit()
```

15.2.15 Preferences

```
PreferencesEdit()
```

15.3 View

15.3.1 Fullscreen

```
if AppShowGet() == 'fullscreen':
    AppShowSet('normal')
else:
    AppShowSet('fullscreen')
```

15.4 Data

15.4.1 Sort Spreadsheet

```
eval( SpreadsheetPromptSort() or 'None' )
```

15.4.2 Filter Spreadsheet

```
eval( SpreadsheetPromptFilter() or 'None' )
```

15.4.3 Create Column

```
eval( SpreadsheetPromptColumnExpression() or 'None' )
```

15.4.4 Create Empty Column

```
_tmp_col   = PromptString("Column Name")
_tmp_table = DataGetTable( DatatableCurrent() )
if _tmp_table.GetColumn( _tmp_col ) > _tmp_table.NumCols():
    DatatableSetExpression( DatatableCurrent(), _tmp_col, "" )
else:
    PromptError("Column %s already exists"%repr(_tmp_col))
```

15.4.5 Format Columns

```
for x in SpreadsheetPromptFormat()[:-1].split(';'):
    eval(x or 'None')
```

15.4.6 Organize Columns

```
SpreadsheetColumnController()
```

15.4.7 Advanced Depiction Options

```
SpreadsheetPromptGraphemeOpts()
```

15.4.8 Show Statistics

```
SpreadsheetShowStatsSet(SpreadsheetCurrent(), \
                        not SpreadsheetShowStatsGet(SpreadsheetCurrent()))
```

15.5 Style

15.5.1 Color

Palette

```
try:
    _tmp_color = PromptColor()
    ColorSetScoped(_tmp_color)
except:
    pass
```

Unique

```
ColorUniqueScoped()
```

Restore

```
MoleculeColorsResetScoped()
```

Molecules

All but two of the options (*Carbon* and *Reference*) in this menu call the following function.

```
MoleculeColorByScoped(SCHEME)
```

The SCHEME parameter is assigned based on which menu option was selected according to the table below.

Table 1: Color schemes

Amino	'amino'
Alternate Locations	'alt location'
BFactor	'bfactor'
Chain	'chain'
CPK	'cpk'
CPKNew	'cpknew'
Element	'element'
Formal Charge	'formal charge'
Group	'group'
Partial Charge	'partial charge'
Residue	'residue'
Shapely	'shapely'

The *Carbon* menu option calls the following function.

```
try:
    _tmp_color = PromptColor()
    AtomColorSetScoped(6, _tmp_color)
except:
    pass
```

The *Reference* menu option calls the following function.

```
AtomColorReferenceScoped()
```

Surfaces

All of the options in this menu call the following function.

```
SurfaceColorByScoped(SCHEME, PARAMS)
```

Both the SCHEME and PARAMS parameters are assigned based on which option was selected according to the table below.

Table 2: Color schemes and parameters

Atom Color	'atom'	''
Curvature	'curvature'	''
Distance	'distance'	''
Electrostatic	'electrostatics'	''
Hydrogen Bond Potential	'hydrogen bonds'	''
Hydrophobicity	'hydrophobicity'	''
Surface Potential	'potential'	''

The *Grid* menu options calls the following function.

```
try:
    id = PromptID(['g'], False)
    SurfaceColorByScoped('grid', str(id))
except:
    pass
```

Transparency

All of the options in this menu call the following function.

```
SurfaceTransparencySetScoped(N)
```

The N parameter is assigned to a value between 0 and 255 based on which option was selected. A value of 255 corresponds to 0 percent transparency and a value of 0 corresponds with 100 percent transparency. The specific values used in the menu are: 255, 230, 204, 179, 153, 128, 102, 77, 61, 26, 0.

15.5.2 Selection

Crop Selected

```
SurfaceCropScribedScoped()
```

Crop Unselected

```
SurfaceCropUnscribedScoped()
```

15.5.3 Display

Molecules

All of the options in this menu call the following function.

```
MoleculeStyleSetScoped(STYLE)
```

The STYLE parameter is assigned based on which option was selected according to the table below.

Table 3: Molecule Styles

Wireframe	'wireframe'
Stick	'stick'
Ball and Stick	'ball and stick'
CPK	'cpk'
Stars	'stars'
Hidden	'hidden'

Hydrogens

All of the options in this menu call the following function.

```
AtomHydrogenStyleSetScoped(STYLE)
```

The STYLE parameter is assigned based on which option was selected according to the table below.

Table 4: Hydrogen Styles

All	2
Polar	1
None	0

Non-Bonded Atoms

Both of the options in this menu call the following function with the HIDE parameter assigned based on which option was selected (*Show* assigns HIDE to False and *Hide* assigns HIDE to True).

```
BondHideNonbondedSet(HIDE)
```

C-Alpha Trace

Both of the options in this menu call the following function with the DRAW parameter assigned based on which option was selected (*Show* assigns DRAW to True and *Hide* assigns DRAW to False).

```
DrawCATracesSetScoped(DRAW)
```

Ribbons

Both of the options in this menu call the following function with the DRAW parameter assigned based on which option was selected (*Show*} assigns DRAW to True and *Hide*} assigns DRAW to False).

```
DrawRibbonsSetScoped(DRAW)
```

Atom Labels

The *Atom Labels* submenu provides a variety of labeling of options. The "Custom…" option launches a dialog which allows the user to build up more complex and customized labels. This dialog is launched using the following Python command:

```
ViewerLabelDialog('Atoms')
```

All of the other options in the submenu all call the following Python command but with different string arguments:

```
AtomLabelSetScoped('')
```

The arguments used can be found in the table below:

Table 5: Atom Label Arguments

Name	'%n'
Index	'%i'
Element	'%e'
Isotope	'%iso'
Type	'%t'
Integer Type	'%it'
Implicit H Count	'%h'
Map Idx	'%m'
Partial Charge	'%p'
Formal Charge	'%f'
Atom Radius	'%rad'
Residue Info	'%r'
Residue Info (CA)	'%car'
B-Factor	'%bf'
C-Alpha B-Factor	'%cab'
Degree	'%d'
Is Chiral	'%c'
Chirality	'%chi'
Hybridization	'%hy'
Has Stereo Specified	'%sts'
Is In Ring	'%rng'
OEChem Idx	'%oei'
Symmetry Class	'%sy'
Atom Comment	'%cm'
Atom Occupancy	'%occ'
MMFF Atom Type	'%mmff'
Generic Data	'%gd(<GENERIC_DATA_TAG>)'
None	''

Bond Labels

The *Bond Labels* submenu provides a variety of labeling of options. The "Custom…" option launches a dialog which allows the user to build up more complex and customized labels. This dialog is launched using the following Python command:

```
ViewerLabelDialog('Bonds')
```

All of the other options in the submenu all call the following Python command but with different string arguments:

```
BondLabelSetScoped('')
```

The arguments used can be found in the table below:

Table 6: Bond Label Arguments

Name	'%n'
Index	'%i'
Type	'%t'
Integer Type	'%it'
Order	'%o'
Length	'%l'
Is Chiral	'%c'
Chirality	'%chi'
Has Stereo Specified	'%sts'
OEChem Idx	'%oei'
Generic Data	'%gd(<GENERIC_DATA_TAG>)'
None	''

Electrostatic Potential

Both of the options in this menu call the following function with the DRAW parameter assigned based on which option was selected (*Show* assigns DRAW to True and *Hide* assigns DRAW to False).

```
ShowESGridScoped(DRAW)
```

Molecular Surface

Both of the options in this menu call the following function with the DRAW parameter assigned based on which option was selected (*Show* assigns DRAW to True and *Hide* assigns DRAW to False).

```
ShowSurfaceScoped('molecular', DRAW)
```

Accessible Surface

Both of the options in this menu call the following function with the DRAW parameter assigned based on which option was selected (*Show* assigns DRAW to True and *Hide* assigns DRAW to False).

```
ShowSurfaceScoped('accessible', DRAW)
```

Hydrogen Bonds

Add Targets

```
HBondAddTargetsScoped()
```

Remove Targets

```
HBondRemoveTargetsScoped()
```

External Hydrogen Bonds

```
HBondShowExternalSet(not HBondShowExternalGet())
```

Internal Hydrogen Bonds

```
HBondShowInternalSet(not HBondShowInternalGet())
```

Grids

Solid

```
ContourDrawAsSurfaceSetScoped(True)
```

Mesh

```
ContourDrawAsSurfaceSetScoped(False)
```

Cloud

```
ContourDrawStyleSetScoped('cloud')
```

Grid Corners

```
GridShowCornersSet(not GridShowCornersGet())
```

Electrostatic

```
GridTypeSetScoped(1)
```

ET

```
GridTypeSetScoped(5)
```

FRED

```
GridTypeSetScoped(2)
```

Generic

```
GridTypeSetScoped(0)
```

Difference Map

```
GridTypeSetScoped(4)
```

Regular Map

```
GridTypeSetScoped(3)
```

Add Contour

```
ContourTypedAddScoped()
```

Delete Contour

```
ContourTypedRemoveScoped()
```

Surfaces

All of the options in this menu except one (*Restore*) call the following function.

```
SurfaceStyleSetScoped(STYLE)
```

The STYLE parameter is assigned based on which option was selected according to the table below.

Table 7: Surface Style Arguments

Solid	'solid'
Mesh	'mesh'
Points	'points'

The *Restore* option calls the following function.

```
SurfaceRestoreScoped()
```

15.5.4 Graphics

Depth Cue

```
ViewerDepthcueSet(not ViewerDepthcueGet())
```

Slabbing

```
ViewerSlabEnableSet(not ViewerSlabEnableGet())
```

Mirror Slabs

```
ViewerMirrorSlabsSet(not ViewerMirrorSlabsGet())
```

Slab Level

All of the options in this menu call the following function.

```
ViewerSlabEnableSet(True)
ViewerSlabFarSet(50+N/2)
```

The N parameter is assigned to a value between 10 and 80 corresponding to the desired percent of the scene to be contained within the slabbing planes.

Render Quality

All of the options in the menu call the following function.

```
ViewerLODSet(LOD)
```

The LOD parameter is assigned based on which option was selected according to the table below.

Table 8: Level of Detail Parameters

Fastest	1
Fast	2
Good	3
Better	4
Best	5
Presentation	6

Note: *Presentation* quality is optimized for creating output images with a white background.

Stereo

Off

```
ViewerStereoEnableSet(False)
```

Splitscreen

```
ViewerStereoEnableSet(True)
ViewerStereoHardwareSet(False)
```

Stencil

```
ViewerStereoStyleSet(2)
ViewerStereoEnableSet(True)
```

Angle

```
try:
    angle = PromptFloat('Stereo Angle')
    ViewerStereoAngleSet(angle)
except:
  pass
```

Separation

```
try:
    sep = PromptFloat('Stereo Separation')
    ViewerStereoSeparationSet(sep)
except:
    pass
```

Cross-Eyed

```
ViewerStereoCrossEyedSet(not ViewerStereoCrossEyedGet())
```

15.5.5 Symmetry

Edit Cell

```
XRayEditCrystalParams(ActiveID())
```

Show Unit Cell

```
DrawUnitCellSet(OEInternal.MenuVal)
```

Show Symmetry

```
DrawSymmetrySet(OEInternal.MenuVal)
```

Realize Symmetry In Cell

```
DoSymmetryRealize(True)
```

Realize Symmetry

```
DoSymmetryRealize(False)
```

Symmetry Radius

```
SymmetryRadiusSet(PromptFloat('Symmetry radius',SymmetryRadiusGet()))
```

Symmetry Operators

No Symmetry Color

```
SymmetryColorHelper(0,OEInternal.MenuVal)
```

Single Symmetry Color

```
SymmetryColorHelper(1,OEInternal.MenuVal)
```

Color By Operator

```
SymmetryColorHelper(2,OEInternal.MenuVal)
```

Unique Color

```
SymmetryColorHelper(3,OEInternal.MenuVal)
```

15.6 Bookmarks

15.6.1 Add

```
try:
    _tmp_name = PromptString('Bookmark name', 'Bookmark ' + str(1 +␣
↪len(Bookmarks())))
    if _tmp_name:
        BookmarkSave(_tmp_name)
except:
    pass
```

15.6.2 Organize

```
exec(BookmarkOrganizeDialog())
```

15.6.3 Animated

```
ViewerBookmarksSetAnimated(not ViewerBookmarksGetAnimated())
```

15.7 Tools

15.7.1 FRED View

```
CustomViewFRED()
```

15.7.2 EON View

```
CustomViewEON()
```

15.7.3 ROCS View

```
CustomViewROCS()
```

15.8 Select

The Select menu provides a number of ways to select groups of atoms for use in various tasks, like coloring atoms, creating surfaces, etc. Most selection operations involve only molecules which are visible in the 3D Window. The Select menu also provides the ability to name sets of atoms and store them for later retrieval. These sets can be combined with the current selection through logical operations, allowing a high degree of flexibility in creating selection sets.

15.8.1 Clear

Clears all selected atoms and bonds.

15.8.2 All

Selects all visible atoms and bonds.

15.8.3 Invert

Inverts the current selection with respect to visible atoms.

15.8.4 Protein

Operations in this menu are protein-centric.

Backbone

Selects backbone atoms only.

Sidechains

Selects side chain atoms.

C-Alphas

Selects alpha carbon atoms in a protein.

Residue #...

Allows selection of atoms by residue number. A dialog will prompt for residue numbers and chain ID's to include in the selection.

Residue Type...

Selects standard amino and nucleic acid residues by type. A dialog will prompt for residue names to be included in the selection.

Complete Residues

Selects all atoms in any residue that has at least one atom selected.

Zero Occupancy

Selects all atoms with an occupancy value of zero.

Partial Occupancy

Selects all atoms with an occupancy value less than 1.0.

Alt Locations

Selects all residues with alternate location specifiers.

Waters

Selects all water molecules.

15.8.5 Atom Numbers...

Selects atoms by their OEChem index value. A dialog will prompt for the atom numbers to include in the selection.

15.8.6 By Distance...

Selects residues within a specified distance of the currently selected atoms. A dialog will appear to allow the distance to be specified with a slider. A checkbox in this dialog provides an option to exclude the originally selected atoms from the final selection.

15.8.7 Heteroatoms

Selects all atoms that are not carbon or hydrogen.

15.8.8 Charged

Selects all atoms with a non-zero formal charge.

15.8.9 SMARTS Pattern...

Selects all atoms matching a SMARTS query pattern. A dialog will prompt for the SMARTS pattern to use.

15.8.10 Expression...

Selects atoms using an expression from VIDA's selection language. A dialog will prompt for the expression to use.

15.8.11 Store

These options allow the storage of the current selection into a new or existing atom set.

New Set...

Creates a new atom set after prompting for a set name.

Into <Set_Name>

Replaces the contents of the named set *Set_Name* with the current selection.

15.8.12 Recall

From <Set_Name>

Selects the atoms in the named set *Set_Name*.

15.8.13 Combine

Combines the current selection with named sets.

Include

Adds the atoms from the named set to the current selection (a logical OR operation).

Exclude

Removes atoms from the named set from the current selection (logical NOT operation).

Intersect

Selects the atoms that are in both the named set and the current selection (logical AND operation).

15.8.14 Delete Set

Deletes the named set. This does not delete any atoms.

15.8.15 Hide Selected

Turns off the visibility of all selected atoms.

15.8.16 Hide Unselected

Turns off the visibility of all unselected atoms in visible molecules.

15.9 Window

15.9.1 Main Window

2D Viewer

```
    if OEInternal.MenuVal:
        AppMainWindowSet('2D Viewer')
```

3D Viewer

```
    if OEInternal.MenuVal:
        AppMainWindowSet('3D Viewer')
```

Spreadsheet

```
    if OEInternal.MenuVal:
        AppMainWindowSet('Spreadsheet')
```

15.9.2 2D Preview

```
    WindowVisibleSet('2D Preview')
```

15.9.3 2D Viewer

```
    WindowVisibleSet('2D Viewer')
```

15.9.4 3D Viewer

```
    WindowVisibleSet('3D Viewer')
```

15.9.5 List Window

```
    WindowVisibleSet('List Window')
```

15.9.6 Process Manager

```
    WindowVisibleSet('Process Manager')
```

15.9.7 Scripting Window

```
    WindowVisibleSet('Scripting Window')
```

15.9.8 Spreadsheet

```
    WindowVisibleSet('Spreadsheet')
```

15.9.9 Style Control

```
    WindowVisibleSet('Style Control')
```

15.10 Help

15.10.1 About

```
About()
```

15.10.2 Documentation

All of the items in this menu open a specific document. The script to open the document can be found below:

```
AppOpenUrl('file:' + AppDocDir() + FILE_PATH)
```

The actual document path that is used is:

- *User Manual (HTML) <https://docs.eyesopen.com/applications/vida/index.html>_.*

15.10.3 License

Open

```
AppOpenUrl('file:' + AppLicenseFile())
```

Set

```
AppLicenseUpdate(True)
```

Update

```
AppLicenseUpdate(False)
```

15.10.4 Open Examples Directory

```
AppOpenUrl('file:' + AppExamplesDir())
```

15.10.5 Open User Directory

```
AppOpenUrl('file:' + AppUserDir())
```

15.10.6 File Bug Report

```
AppOpenUrl('mailto:support@eyesopen.com')
```

CHAPTER

SIXTEEN

EXAMPLES

VIDA ships with a large collection of example data as well as example scripts. The examples are contained in the "examples" directory in the VIDA installation. This directory can be directly accessed from within VIDA by selecting the "Open Examples Directory" option in the "Help" menu.

A fast way to get started exploring the example data and scripts is to open the `add_menu_updating.vpy` script (see *Opening Files*) which will add an "Examples" menu to the main menu bar. This new menu will contain an entry for each example data file and script provided. Selecting any one of these options will open the associated file in the current instance of VIDA.

16.1 Data

VIDA ships with a directory of example data for the user to experiment with in the event that other input files are not immediately available. Included in this directory are:

- `cox2_good.sdf.gz` - This is a large set of Cox-2 inhibitors which contains associated molecular data which makes this an excellent example file for use in the spreadsheet.
- `MMFF94.mol2.gz` - This is another large set of small molecules but it contains a significantly greater amount of diversity among the molecules than the previous file. This file is an excellent example file for browsing through a large data set as well as for exploring the many different ways that shape and chemistry may be visualized with VIDA.
- `1ll5.grd` - This is a grid file containing electron density data corresponding to crystallography data of the ligand structure found in the following file. It provides an excellent example of VIDA's grid visualization capabilities and also works well to provide a context for the associated ligand file.
- `1ll5.lig.pdb.gz` - This small molecule is a beta-lactamase inhibitor positioned in its crystallographically determined binding location. This file combined with the actual beta-lactamase protein found in the following file provide an excellent example of how VIDA can be used to visualize small molecule-protein interactions. When visualizing this molecule, be sure to observe the three excellent hydrogen bonds it makes with the protein.
- `1ll5.nolig.pdb.gz` - This file contains the crystallographically determined structure of a beta-lactamase protein. Its bound inhibitor can be found in the previous file. This file is an excellent example of the many ways that proteins can be visualized in VIDA. Be sure to observe its many internal hydrogen bonds, particularly in the active site.
- `1ll5.omega.oeb.gz` - This file contains a large number of conformations of the beta-lactamase ligand (`1ll5.lig.pdb.gz`) generated with OpenEye's conformer generation program *OMEGA*. This file is an excellent example of VIDA's ability to handle multi-conformer molecules as well as its ability to view *OMEGA* output. It is important to note that this file is not in the same coordinate space as the three previous files and as such should not be expected to appear in the same location on screen with the above files.

16.2 Scripts

VIDA ships with a large directory of example scripts to both highlight the many different aspects of VIDA's scripting capability and to help guide the user to begin developing their own scripts. Each one of the example scripts provided will be discussed in detail in the following sections. Note that not all of the example scripts will actually do anything interesting when you load them directly into VIDA. Many of them illustrate the creation of useful functions, but may not actually call these functions.

The location of the example scripts is platform-specific, but can be found from within VIDA by selecting the "Open Examples Directory" option in the "Help" menu.

16.2.1 add_menu.vpy

This example shows how to add a new menu to the top level main menu (if it does not already exist). Note that the call to `MenuAddSubmenu` returns the menu name to be used to reference that menu in subsequent functions. It is important to use the returned name to avoid potential naming clashes when accessing the menu.

```
menu_name = "Examples"
if not MenuExists(menu_name):
  menu_name = MenuAddSubmenu("MenuBar", menu_name, "Examples", "", False)
```

Once the top level menu has been created, this example shows how to add submenus to the newly created menu.

```
data_menu_name = "Example_Data"
if not MenuExists(data_menu_name):
  data_menu_name = MenuAddSubmenu(menu_name, data_menu_name, "Data", "", False)

scripts_menu_name = "Example_Scripts"
if not MenuExists(scripts_menu_name):
  scripts_menu_name = MenuAddSubmenu(menu_name, scripts_menu_name, "Scripts", "", False)
```

Lastly, this example details how to populate the newly created submenus with the relevant items and their associated Python commands. In this case, the script looks for the *Examples* directory that ships with VIDA and then scans that directory for example data and scripts which are then referenced in the appropriate submenus.

```
import os

example_dir = AppExampleDir()
if example_dir and os.path.exists(example_dir):
  example_data_dir = os.path.join(example_dir, "data")
  if os.path.exists(example_data_dir):
    for file in os.listdir(example_data_dir):
      fullpath = os.path.abspath(os.path.join(example_data_dir, file))
      if os.path.isfile(fullpath):
        command = "Open(" + repr(fullpath) + ")"
        MenuAddButton(data_menu_name, file, command)

  example_scripts_dir = os.path.join(example_dir, "scripts")
  if os.path.exists(example_scripts_dir):
    for file in os.listdir(example_scripts_dir):
      fullpath = os.path.abspath(os.path.join(example_scripts_dir, file))
      if os.path.isfile(fullpath):
        if ".py" in file or ".vpy" in file or ".pyv" in file:
          command = "execfile(" + repr(fullpath) + ")"
          MenuAddButton(scripts_menu_name, file, command)
```

16.2.2 add_menu_item.vpy

This example shows how to add a new entry to an already existing menu. The new entry when clicked on in the menu will execute the specified Python command.

```
menu_name    = "Help"
menu_item    = "Subscribe To OpenEye"
menu_command = 'AppOpenUrl("https://eyesopen.com/subscribe")'

MenuAddButton(menu_name, menu_item, menu_command)
```

16.2.3 add_menu_updating.vpy

This example shows how to create a new menu which is populated by a specified Python command each time the menu is shown. In this example, each time the user clicks on the *Examples* menu, the specified `UpdateExamplesMenu` function will be called before the menu is actually shown.

```
menu_name = "Examples"
if not MenuExists(menu_name):
  menu_name = MenuAddSubmenu("MenuBar", menu_name, "Examples", "UpdateExamplesMenu()",␣
↪False)
```

The `UpdateExamplesMenu` function is defined below. The first thing this function does is to clear the previous contents of the menu by calling the `MenuRemoveAll` function. It then searches for the *Examples* directory that is provided by VIDA and then scans that directory for example data and scripts which are then referenced in the appropriate submenus.

```

```

The process by which the menus are created and populated here is somewhat different from the *add_menu.vpy* example due to the dynamic nature of the menu creation in this example.

16.2.4 add_toolbar_button.vpy

This example shows how to add a simple button to the main application toolbar which when pressed will execute the specified Python command (in this case *DeleteAll*).

```
icon    = "cancl_32"
command = "DeleteAll()"
tooltip = "Delete everything and start over"
toolbar = "Application"
name    = "ToolbarClearButton"

try:
    ToolbarRemove(toolbar, name)
except:
    pass

ToolbarAddViaNamedIcon( icon, command, tooltip, toolbar, name )
```

It is worth nothing that `ToolbarRemove` is called before `ToolbarAddViaNamedIcon` in order to ensure that multiple buttons will not be created and thus fill up the toolbar if the script is run multiple times.

16.2.5 add_toolbar_toggle.vpy

This example shows how to add a simple toggle button to the main application toolbar. A *toggle* button is different from a standard *push* button in that it has two potential states and can display a different icon and execute different Python commands based on the current state.

```
onIcon     = "mol_32"
offIcon    = "depict_32"
onCommand  = "AppMainWindowSet('2D Viewer');WindowVisibleSet('3D Viewer', False)"
offCommand = "AppMainWindowSet('3D Viewer');WindowVisibleSet('2D Viewer', False)"
tooltip    = "Toggles the main window between the 2D and 3D viewers"
update     = "AppMainWindowGet() == '2D Viewer'"
toolbar    = "Application"
name       = "ToolbarToggleMainWindowButton"

try:
    ToolbarRemove(toolbar, name)
except:
    pass

ToolbarAddToggleViaNamedIcons( onIcon, offIcon, onCommand, offCommand, tooltip, \
                               update, toolbar, name )
```

As seen in the example above, a toggle button also supports specification of an *update* command which may be executed to determine the current state of the button. The specified *update* code is normally executed when a call to `ToolbarUpdate` or `ToolbarItemUpdate` is made.

```
def AppMainWindowSet( windowName, _oldAppMainWindowSet = AppMainWindowSet ):
  _oldAppMainWindowSet(windowName)
  ToolbarItemUpdate(name, toolbar)
```

To ensure that this particular toggle button is kept in sync with the state of application, the `AppMainWindowSet` function is overridden to include a call to `ToolbarItemUpdate`. This is particularly important as there are other common pathways within VIDA which can have the same effect as toggling this button (such as changing the main window through the *Windows* main menu).

16.2.6 calc_et.vpy

This example shows how to create a function using the **OEChem** and **Zap** Python toolkits from OpenEye to calculate the electrostatic Tanimoto between two molecules currently loaded in VIDA.

```
from openeye import oechem
from openeye import oezap

def CalculateET( idOrKey1, idOrKey2 ):
  tanimoto = 0.0

  try:
    PushIgnoreHint(True)
    WaitBegin()

    refmol = MoleculeGet(idOrKey1)
    oechem.OEMMFFAtomTypes(refmol)
    oechem.OEMMFF94PartialCharges(refmol)
```

(continues on next page)

(continued from previous page)

```
    oechem.OEAssignBondiVdWRadii(refmol)

    fitmol = MoleculeGet(idOrKey2)
    oechem.OEMMFFAtomTypes(fitmol)
    oechem.OEMMFF94PartialCharges(fitmol)
    oechem.OEAssignBondiVdWRadii(fitmol)

    et = oezap.OEET()
    et.SetRefMol(refmol)
    tanimoto = et.Tanimoto(fitmol)

  finally:
    WaitEnd()
    PopIgnoreHint()

  return tanimoto
```

The `CalculateET` function retrieves copies of the specified molecules from VIDA, prepares them for the calculation and then performs the actual calculation. The resulting Tanimoto value is then returned by the function.

```
def CalculateETsForList( id ):
  ids = ListGetObjects(id)
  if len(ids) > 1:
    tanimoto = 0.0
    try:
      PushIgnoreHint(True)
      WaitBegin()

      refmol = MoleculeGet(ids[0])
      oechem.OEMMFFAtomTypes(refmol)
      oechem.OEMMFF94PartialCharges(refmol)
      oechem.OEAssignBondiVdWRadii(refmol)

      et = oezap.OEET()
      et.SetRefMol(refmol)

      count = 0
      total = len(ids)-1
      for id in ids[1:]:
        fitmol = MoleculeGet(id)
        oechem.OEMMFFAtomTypes(fitmol)
        oechem.OEMMFF94PartialCharges(fitmol)
        oechem.OEAssignBondiVdWRadii(fitmol)
        tanimoto = et.Tanimoto(fitmol)

        DataAdd(KeysGet(id)[0], "ET", str(tanimoto))

        ProgressbarUpdate(count, total)
        count = count + 1

    finally:
```

(continues on next page)

(continued from previous page)

```
    ProgressbarUpdate(0, 0)
    WaitEnd()
    PopIgnoreHint()
```

The `CalculateETForList` function expects a VIDA list ID as a parameter. The function will then assign the first molecule in the list to be the reference structure against which the ET will be calculated for all the other molecules in the list.

Instead of returning a list of ET values, the calculated ET is stored as a data field in the application spreadsheet using the `DataAdd` function.

16.2.7 calculate_clashes.vpy

This example shows how to use the **OEChem** toolkit within VIDA to detect and then visualize intermolecular clashes.

```
from openeye import oechem

def CalculateClashes( proteinIDOrKey, ligandIDOrKey, cutoff ):
  protein = MoleculeGet(proteinIDOrKey)
  ligand  = MoleculeGet(ligandIDOrKey)

  if protein and ligand:
    try:
      PushIgnoreHint(True)
      WaitBegin()

      nbrs = oechem.OEGetNearestNbrs(protein, ligand, cutoff)
      for nbr in nbrs:
        id = MonitorDistanceCreate(nbr.GetBgn().GetKey(), nbr.GetEnd().GetKey())
        if id:
          mkey = KeysGet(id)[0]
          MonitorColorSet(mkey, OEColor(255, 0, 0))

    finally:
      PopIgnoreHint()
      WaitEnd()
```

The `CalculateClashes` function takes the IDs of two molecules and a cutoff distance used to determine the intermolecular clashes. The clashes are determined by calling the **OEChem** function `OEGetNearestNbrs`. Any clashes that are found are then visualized as distance monitors (colored red) between the clashing atoms.

16.2.8 cluster_browsing.vpy

This example is one of the most complicated examples included in this set as it incorporates many aspects of VIDA scripting into a single example. This example shows how one can create their own custom cluster browsing interface within VIDA using clustering information stored in the SD data of the relevant molecules.

The first part of the example attempts to locate an associated '.ui' file which defines the layout of the widget used in this example. The '.ui' file is expected to be located in the same directory as the script. If the '.ui' file is not found, the script will throw an exception.

```
import os
```

(continues on next page)

(continued from previous page)

```
path   = AppExampleDir()
uifile = os.path.join(os.path.join(path, "scripts"), "cluster_browsing.ui")
if not os.path.exists(uifile):
  raise IOError("Unable to locate user-interface specification: '%s'" % uifile)
```

A picture of the widget template defined by the '.ui' file can be seen in the figure *below*. This widget was created using the *Qt Designer* application and will be loaded into an actual class using PySide2.

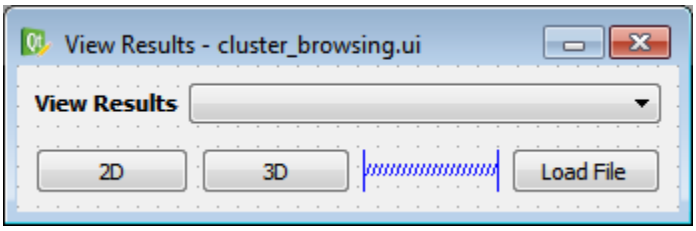


Fig. 1: **cluster_browsing Widget Template**

The actual widget class is created below using PySide2. The class constructor performs the required initializations, loads the ui file, and sets up the necessary connections between the internal widgets and the classes methods.

```
from openeye           import vfappqt
from PySide2.QtCore    import *
from PySide2.QtWidgets import *
from PySide2           import QtUiTools

class ClusterBrowser(QWidget):

  def __init__(self):
    QWidget.__init__(self)
    self.setWindowTitle("View Results")

    # load the ui file and put it into a layout

    self.layout = QVBoxLayout()
    self.widget = QtUiTools.QUiLoader().load(uifile)
    self.layout.addWidget(self.widget)

    self.setContentsMargins(0, 0, 0, 0)
    self.setLayout(self.layout)

    # wrap the widget so we can get to the sub-widgets.
    # this uses the same names as set in Designer

    ui = self.ui = vfappqt.WrapPyQt(self.widget)

    # connect the buttons

    self.connect(ui.buttonLoad, SIGNAL("clicked()"), self.loadFile)
    self.connect(ui.comboGroups, SIGNAL("currentIndexChanged(int)"), self.changeGroup)
    self.connect(ui.button2D, SIGNAL("clicked()"), self.show2D)
    self.connect(ui.button3D, SIGNAL("clicked()"), self.show3D)
```

(continues on next page)

(continued from previous page)

```
  def loadFile(self):
    cmd = "ViewClusterFile(PromptFilename('', 'Open', 'molecule', 'Open clustered results
↪'))"
    exec(cmd)

  def changeGroup(self, index):
    data = str(self.ui.comboGroups.itemData(index).toString())
    exec(data)

  def show2D(self):
    AppMainWindowSet("2D Viewer")

  def show3D(self):
    AppMainWindowSet("3D Viewer")
```

Once the `ClusterBrowser` widget class has been defined, an actual instance of that class needs to be created. In order to ensure that the `ClusterBrowser` can be embedded into frame around the main window, a `QDockWidget` is created to contain the `ClusterBrowser` and then added to the right-hand side of the main window.

In addition, a menu item is added to the *Tools* menu which will prompt the user to specify an input file and then will show the `ClusterBrowser` widget.

```
widget = ClusterBrowser()
dock   = QDockWidget("Cluster Browser")
dock.setObjectName("Cluster Browser")
dock.setWidget(widget)
dock.layout().setContentsMargins(0,0,0,0)

for tw in QApplication.topLevelWidgets():
  if tw.inherits("QMainWindow"):
    tw.addDockWidget(Qt.RightDockWidgetArea, dock)
    break

dock.hide()

MenuAddButton("Tools", "Cluster View...",
                        "ViewClusterFile(PromptFilename('', 'Open', 'molecule', 'Open␣
↪clustered results'))")
```

As described above, the molecules are expected to contain cluster information in their associated SD data. This data can be viewed and processed in the spreadsheet. To help organize this information, a custom sorting function is defined below.

```
def Sort(name):
  SpreadsheetHideColumn(name, "Depiction")
  if name=="Ideas":
    SpreadsheetSort(name, ["Idea"], [1,], False)
  else:
    SpreadsheetSort(name, ["VIDA ID"], [2,], False)
```

The `ViewClustersID` function defined below takes as a parameter the ID of a list of molecules already loaded into VIDA which contain the appropriate SD data. This function then creates a new filtered spreadsheet called "Ideas" which only contains the cluster centroids (or "Ideas"). For each "Idea" there is an associated "Idea Group" of other

molecules. A new filtered spreadsheet is also created for each individual "Idea Group", although it is hidden from the main spreadsheet tab by default.

This function also populates the drop-down menu in the `ClusterBrowser` widget defined above with an entry for each of the "Idea Groups".

```
maxGroups  = 0
ideas      = []
comboName  = ""

def ViewClustersID( id ):
  global comboName
  global ideas
  global maxGroups

  if not IsAList(id):
    return

  ids = ListGetObjects(id)
  if ids:
    static = True

    try:
      PushIgnoreHint(True)

      SpreadsheetFilter("Molecules",
                                              "(COL('Idea')==0 or COL('Idea')) and COL(
↪'IdeaGroup')==COL('Idea')",
                                              "Ideas", static)
      Sort("Ideas")
      SpreadsheetCurrentSet("Ideas")
      ct = SpreadsheetGetNumRows("Ideas")
      counts = [0]*(ct+1)
      ideas = [OEKey()]*(ct+1)
      for i in range(0, ct):
        try:
          group = int(SpreadsheetData("Ideas", i, "IdeaGroup"))
          ideas[group] = SpreadsheetGetKeyForRow("Ideas", i)
        except(Exception):
          pass

      for i in range(0, SpreadsheetGetNumRows("Molecules")):
        groupVal = SpreadsheetData("Molecules", i, "IdeaGroup")
        try:
          group = int(groupVal)
          counts[group] += 1
        except(Exception):
          pass

      AppStatusTextSet("Parsing Idea Groups", 0)

      maxGroups  = SpreadsheetGetNumRows("Ideas")
      groupNames = [ "None", "Ideas", ]
      groupCmds  = [ "ViewIdeaGroup(-2)", "ViewIdeaGroup(-1)", ]
```

(continues on next page)

(continued from previous page)

```
    for i in range(0, maxGroups):
      if counts[i] > 0:
        groupNames.append("Idea Group %d (%d)" % (i, counts[i]))
        groupCmds.append("ViewIdeaGroup(%d)" % i)

    SpreadsheetCurrentSet("Ideas")
    ClearVisible()
    ClearActive()
    ClearSelection()

    Active(ideas[0])
    Visible(ideas[0], True)
    DrawMatrixSet(True)

    widget.ui.comboGroups.clear()
    for (name, cmd) in zip(groupNames, groupCmds):
      widget.ui.comboGroups.addItem(name, QVariant(cmd))
    dock.show()

  finally:
    PopIgnoreHint()
```

The `ViewClusterFile` function loads a specified molecule file and then passes the ID of the newly created list to the `ViewClustersID` function to setup the view. Before calling this function, the spreadsheet window is hidden to prevent flickering as many spreadsheet operations take place in the `ViewClustersID` function.

```
def ViewClusterFile( fname ):
  try:
    WaitBegin()

    ids = []
    if os.path.exists(fname):
      ids = Open(fname)
      if ids:
        WindowVisibleSet("Spreadsheet", False)
        ViewClustersID(ids[0]-1)
        WindowVisibleSet("Spreadsheet", True)

  finally:
    WaitEnd()
```

The `ViewIdeaGroup` function takes the index of an "Idea Group" as a parameter. If the idea group exists, the associated spreadsheet will be shown. In addition, the first 12 molecules in the "Idea Group" are displayed in tiled mode in the main window (either 2D or 3D, depending on which option the user has selected).

```
lastGroup = ""

def ViewIdeaGroup( group ):
  global lastGroup

  try:
```

(continues on next page)

(continued from previous page)

```
    PushIgnoreHint(True)
    DrawMatrixSet(True)

    ClearActive()
    ClearVisible()
    ClearSelection()

    if lastGroup != "Ideas":
      SpreadsheetHideTab(lastGroup)


    if group >= -1:
      if group == -1:
        lastGroup = "Ideas"
        group     = 0
      else:
        grpQuery = "COL('IdeaGroup')==%d" % int(group)
        lastGroup  = "Grp %d" % int(group)
        SpreadsheetFilter("Molecules", grpQuery, lastGroup, True)
        Sort(lastGroup)
        SpreadsheetShowTab(lastGroup)

      SpreadsheetCurrentSet(lastGroup)
      ct = SpreadsheetGetNumRows(lastGroup)
      if (ct > 12):
        ct = 12

      Active(ideas[group])
      for i in range(0, ct):
        key = SpreadsheetGetKeyForRow(lastGroup, i)
        Visible(key, True)
    else:
      Active(ideas[0])

  finally:
    PopIgnoreHint()
```

16.2.9 docking_view.vpy

This example shows how to create a function which replicates the behavior of the `FRED View` tool available under the *Tools* menu. The function takes a protein ID, a ligand ID and an optional list of IDs corresponding to docking results.

```
def DockingView():
  ligand = ActiveKey()
  marked = GetMarkedMolecules()
  protein = OEKey()
  if marked:
    protein = marked[0]

  if not protein.IsValid() or not ligand.IsValid():
    PromptMessage("""To create a docking view, set the desired protein to
    'marked' in the List Window, and make the ligand 'active' before calling
```

(continues on next page)

(continued from previous page)

```
    this function.""")
    return

  showRibbons   = True
  showLabels    = True
  showHBonds    = True
  showSurface   = True
  residueWithin = 5.0
  surfaceWithin = 4.5
  surfColor     = "electrostatics"

  try:
    PushIgnoreHint(True);
    WaitBegin();

    ClearActive()
    ClearLocked()
    ClearVisible()

    DrawMatrixSet(False)
    BondHideHydrogenSet(True)

    Visible(protein, True)
    Visible(ligand, True)

    Lock(protein, True)
    Lock(ligand,  True)

    Select(ligand, True)
    AtomColorReferenceScoped(SelectedScope)

    SelectWithin(residueWithin, True)
    SelectInvert()

    AtomStyleSetScoped("hidden", SelectedScope)
    BondStyleSetScoped("hidden", SelectedScope)

    DrawRibbonsSet(protein, showRibbons)
    if showLabels:
      AtomLabelSet(protein, "%car")
    else:
      AtomLabelSet(protein, "")

    ClearSelection();
    if showSurface:
      surfID = SurfaceCreate("molecular", protein, True)
      if surfID:
        surfKey = KeysGet(surfID)[0]
        Select(surfKey, True)
        SurfaceColorByScoped(surfColor.lower(), "", SelectedScope)
        SurfaceTransparencySetScoped(128, SelectedScope)
        ClearSelection()
```

(continues on next page)

(continued from previous page)

```
        SurfaceCropDistanceFrom(surfKey, ligand, surfaceWithin)

    if residueWithin > 3.0:
      residueWithin = residueWithin - 3.0
    ViewerFit([ligand, ], residueWithin)
    ViewerLookAt(ligand, protein)

    if showHBonds:
      ClearSelection()
      Select(protein, True)
      HBondAddTargetsScoped(SelectedScope)
      HBondShowExternalSet(True)
      HBondShowInternalSet(False)

    Visible(protein, True)

  finally:
    ClearSelection()
    WaitEnd()
    PopIgnoreHint()
```

As expected from the FRED View behavior, this function makes the protein-ligand complex visible and centers the view on the ligand in the active site. Furthermore, it adds a surface to the protein, colors and crops the surface to a certain distance from the ligand, and hides residues a certain distance from the ligand. The function also adds residue information labels as well as turns on the display of hydrogen bonds.

16.2.10 fix_file_omega.vpy

This example incorporates widget creation using PySide2 and interaction with the **OmegaTK** toolkit. The example shows how **OmegaTK** can be used to generate new conformations for a molecule within VIDA and furthermore how the user can fix certain portions of the molecule during the conformer generation process.

The first part of the example shows how the widget is created using PySide2 from an associated '.ui' file. The '.ui' file is expected to be located in the same directory as the script. If the '.ui' file is not found, the script will throw an exception. In addition, this part of the example also creates the OEOmega object which will be used later to generate the conformations.

```
import os
from PySide2.QtCore    import *
from PySide2.QtWidgets import *
from PySide2           import QtUiTools
from openeye           import vfappqt
from openeye           import oeomega

omegaWidget = QWidget()
path   = AppExampleDir()
uifile = os.path.join(os.path.join(path, "scripts"), "fix_file_omega.ui")
if os.path.exists(uifile):
  omegaWidget = QtUiTools.QUiLoader().load(uifile)
else:
  raise IOError("Unable to locate user-interface specification: '%s'" % uifile)

ui = vfappqt.WrapPyQt(omegaWidget)
```

(continues on next page)

(continued from previous page)

```
omega = oeomega.OEOmega()
omega.SetRMSThreshold(0)

spinMaxConfs = ui.spinMaxConfs
spinMaxConfs.value = omega.GetMaxConfGen()

spinEWindow = ui.spinEWindow
spinEWindow.value  = omega.GetEnergyWindow()
```

A picture of the widget template defined by the '.ui' file can be seen in the figure *below*.

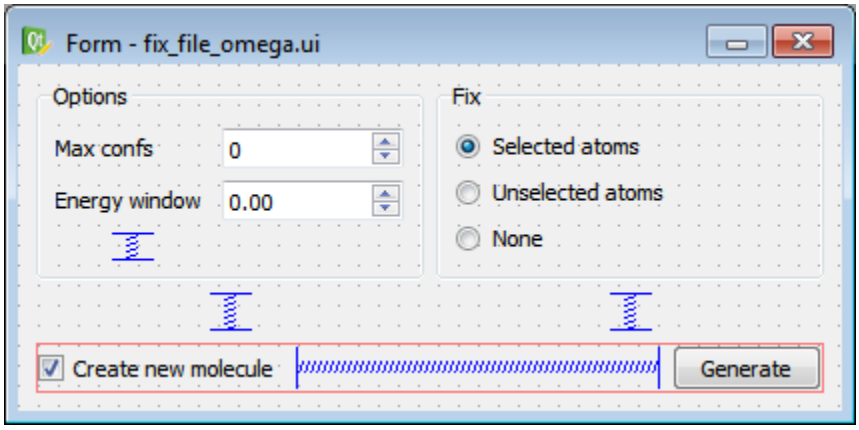


Fig. 2: **fix_file_omega Widget Template**

The following example code details the actual function that will be called to generate the new conformations. It shows how configuration information is pulled from the widget as well as how the fixed portion of the molecule is determined and specified to the Omega object.

```
from openeye import oechem

def GenerateConfs( omega = omega ):
  try:

    AppStatusTextSet("Running omega...", 0)
    WaitBegin()
    try:

      id = ActiveID()
      if not id:
        WaitEnd()
        PromptMessage("A molecule must be focused in order to generate conformers")
        return

      mol = oechem.OEMol()
      MoleculeCheckOut(mol, id)

      omega.SetMaxConfGen(spinMaxConfs.value)
      omega.SetEnergyWindow(spinEWindow.value)
```

(continues on next page)

(continued from previous page)

```
    radioS = ui.radioSelected
    radioU = ui.radioUnselected

    if radioS.isChecked() or radioU.isChecked():
      # assign a parts list, this will end up being 1 if an atom is
      # selected or 0 if not.

      parts = [0] * mol.GetMaxAtomIdx()

      # get all keys from the selected scope

      keys = {}
      for key in GetAtomsByScope(SelectedScope):
        keys[key.key] = True

      for atom in mol.GetAtoms():
        parts[atom.GetIdx()] = atom.GetKey().key in keys

      fixed = oechem.OEPartPredAtom( parts )
      if radioS.isChecked():
        fixed.SelectPart(1)
      else:
        fixed.SelectPart(0)

      fixedMol = oechem.OEMol()
      oechem.OESubsetMol(fixedMol, mol, fixed)

      # check the fixedMolecule to see if it has multiple components
      #  fix-file omega doesn't like this!

      count, components = oechem.OEDetermineComponents( fixedMol )
      if count > 1:
        PromptMessage("Your selection has more than one component, not generating␣
↪conformations...")
        return

      fixedMol.SetTitle("Fixed Portion")
      omega(mol, fixed)

      # attach the fixed molecule to omega conformers so we know
      #  which portions were supposed to be fixed

      mol.SetData("Fixed Portion", fixedMol)

    else:
      omega(mol)

    if ui.checkNewMolecule.isChecked():
      id = MoleculeAdd(mol)
      Active(id)
      ViewerCenterSet(id)
```

(continues on next page)

(continued from previous page)

```
      else:
        MoleculeCheckIn(mol, 0)

    except:
      import traceback, StringIO, sys

      err        = sys.stderr
      sys.stderr = StringIO.StringIO()
      traceback.print_exc()
      print( sys.stderr.getvalue() )
      sys.stderr = err

  finally:
    WaitEnd()
```

This final section of code connects up the function defined above to the appropriate button and then creates a menu item in a 'Special' menu to open the widget.

```
QObject.connect(ui.buttonGenerate, SIGNAL("clicked()"), GenerateConfs)

if not MenuExists('Special'):
  MenuAddSubmenu('MenuBar', 'Special')
MenuAddButton('Special', 'Omega', 'omegaWidget.show()')
```

16.2.11 load_and_split_pdb.vpy

This example shows how to load a file directly from the PDB and then split that file into its individual components. This first section details the function which fetches the molecules corresponding to the specified PDB codes and then loads them into VIDA.

```
import urllib.request

def LoadMoleculesFromPDB( ids ):
  if not ids:
    return

  loaded = []

  WaitBegin()
  try:
    files = []
    for id in ids.split(","):
      id  = id.strip()
      url = urllib.request.urlopen("https://files.rcsb.org/download/" + id + ".pdb.gz")
      if url:
        AppStatusTextSet("Fetching '" + id + "i", 0)
        data = url.read()
        if len(data) > 0:
          f = open(id + ".pdb.gz", "wb")
          f.write(data)
          f.close()
          files.append(id + ".pdb.gz")
```

(continues on next page)

(continued from previous page)

```
      else:
        raise IOError( "PDB id (" + id + ") not found")

  if files:
    loaded = Open(files)

finally:
  WaitEnd()

return loaded
```

The following section details the `LoadAndSplitPDBs` function which calls the `LoadMoleculesFromPDB` function defined above and then iterates over the newly loaded molecules, splits them into their individual components, and then removes the original complex.

```
def LoadAndSplitPDBs( ids ):
  PushIgnoreHint(True)
  WaitBegin()

  try:
    loaded = LoadMoleculesFromPDB( ids )
    for pdb in loaded:
      listid = pdb - 1
      MoleculeNewSubset(listid, pdb, True)
      Delete(pdb)

  finally:
    WaitEnd()
    PopIgnoreHint()
```

This final section shows how a new menu item is created to prompt the user for the desired PDB codes and then to call the above function. This section also shows how to replace the command associated with a menu item in the event that the menu item has already been created.

```
menu    = "Open Special"
name    = "From PDB (Split) ..."
command = "LoadAndSplitPDBs(PromptString('Enter PDB code(s)'))"

if MenuHasItem(menu, name):
  MenuButtonActionSet(menu, name, command)
else:
  MenuAddButton(menu, name, command)
```

16.2.12 molecule_alignment.vpy

This example shows how to create a moderately complex widget using PySide2 to perform molecule alignments using one of a variety of techniques including: shape & color, RMSD, or sequence.

The following block of code shows the function which performs the shape and color alignment of two specified molecules using the **OEChem** and **OEShape** toolkits.

```
from openeye import oechem
from openeye import oeshape

def DoShapeAlign( id1, id2 ):
  refmol = MoleculeGet(id1)
  fitmol = oechem.OEMol()
  MoleculeCheckOut(fitmol, id2)

  best = oeshape.OEBestOverlay()
  best.SetColorForceField(oeshape.OEColorFFType_ImplicitMillsDean)
  best.SetColorOptimize(True)
  if not best.SetRefMol(refmol):
    raise Exception("Unable to set refmol as a reference molecule")

  scoreiter = oeshape.OEBestOverlayScoreIter()
  oeshape.OESortOverlayScores(scoreiter, best.Overlay(fitmol), oeshape.
↪OEHighestTanimotoCombo())
  score = scoreiter.Target()

  fitmol.SetActive(fitmol.GetConf(oechem.OEHasConfIdx(score.fitconfidx)))
  score.Transform(fitmol)

  oechem.OESetSDData(fitmol, "Tanimoto",      "%-.3f" % score.tanimoto)
  oechem.OESetSDData(fitmol, "ColorTanimoto", "%-.3f" % score.GetColorTanimoto())
  oechem.OESetSDData(fitmol, "TanimotoCombo", "%-.3f" % score.GetTanimotoCombo())

  MoleculeCheckIn(fitmol, 0)
  Visible(id2, True)
```

The following block of code shows the function which performs the RMSD alignment of two specified molecules using the **OEChem** toolkit. Note that before the alignment is done, the graphs of the two molecules are compared by the generation of canonical SMILES.

```
from openeye import oechem

def DoRMSDAlign( id1, id2 ):

  refmol = MoleculeGet(id1)
  fitmol = oechem.OEMol()
  MoleculeCheckOut(fitmol, id2)

  smiles1 = oechem.OECreateCanSmiString(refmol)
  smiles2 = oechem.OECreateCanSmiString(fitmol)
  if ( smiles1 != smiles2 ):
    PromptError("Connection table is not the same for refmol and fitmol molecule")
    return

  numconfs     = fitmol.GetMaxConfIdx()
  rmsds        = oechem.OEDoubleArray(numconfs)
  rotations    = oechem.OEDoubleArray(9 * numconfs)
  translations = oechem.OEDoubleArray(3 * numconfs)

  result = oechem.OERMSD(refmol, fitmol, rmsds, True, True, True, rotations,␣
```

(continues on next page)

(continued from previous page)

```
↪translations)
  if not result:
    PromptError("RMSD failed")

  rotations    = list(rotations)
  translations = list(translations)

  indices     = [ (rmsd, index) for index, rmsd in enumerate(rmsds) ]
  rmsd, index = min(indices)

  conformer   = fitmol.GetConf(oechem.OEHasConfIdx(index))
  rotation    = oechem.OEDoubleArray( rotations   [index*9:index*9 + 9] )
  translation = oechem.OEDoubleArray( translations[index*3:index*3 + 3] )

  oechem.OERotate(conformer, rotation)
  oechem.OETranslate(conformer, translation)
  MoleculeCheckIn(fitmol, 0)
  Visible(id2, True)
```

The following block of code shows the function which performs the sequence alignment of two specified proteins using the **OEChem** toolkit. Note that a check for residue information is performed on both molecules to ensure that a sequence alignment is even feasible.

After the alignment is performed in this function, the two molecules are both made visible, uniquely colored and displayed in ribbon style.

```
from openeye import oechem

def DoSequenceAlign( id1, id2 ):
  ref = MoleculeGet(id1)
  fit = oechem.OEMol()
  MoleculeCheckOut(fit, id2)

  for atom in ref.GetAtoms():
    idx = oechem.OEGetResidueIndex(atom)
    if idx >= oechem.OEResidueIndex_ALA and idx <= oechem.OEResidueIndex_VAL:
      break
  else:
    PromptError("Reference molecule has no amino acids")
    return

  for atom in fit.GetAtoms():
    idx = oechem.OEGetResidueIndex(atom)
    if idx >= oechem.OEResidueIndex_ALA and idx <= oechem.OEResidueIndex_VAL:
      break
  else:
    PromptError("Fit molecule has no amino acids")
    return

  sequenceAlignment = oechem.OEGetAlignment(ref, fit)
  rotation          = oechem.OEDoubleArray(9)
  translation       = oechem.OEDoubleArray(3)
```

(continues on next page)

(continued from previous page)

```
  rms = oechem.OERMSD(ref, fit, sequenceAlignment, True, True, rotation, translation)
  oechem.OERotate(fit, rotation)
  oechem.OETranslate(fit, translation)

  try:
    PushIgnoreHint(True)
    MoleculeCheckIn(fit, 0)

    ClearVisible();
    Visible(id1, True)
    Visible(id2, True)
    ViewerCenterSetScoped(VisibleScope, True)
    ViewerFit()
    ColorUniqueScoped(VisibleScope)
    DrawRibbonsSetScoped(True, VisibleScope);
    MoleculeStyleSetScoped("hidden", VisibleScope)

  finally:
    PopIgnoreHint()
```

The following section shows how the main widget is hand-coded (as opposed to using a '.ui' file from *Qt Designer*) using PySide2.

```
from PySide2.QtCore    import *
from PySide2.QtWidgets import *

dock   = QDockWidget("Molecule Alignment")
widget = QWidget()

refmol =  QPushButton('Reference', widget)
refmol.setToolTip('Select a reference molecule to align to')

fitmol = QPushButton('Molecule', widget)
fitmol.setToolTip('Select a molecule that will be aligned')

refmolName = QLabel('Reference ID:', widget)
fitmolName = QLabel('ID:', widget)

refmolID = QLabel('', widget)
fitmolID = QLabel('', widget)

technique = QComboBox(widget)
technique.addItem("shape+color")
technique.addItem("sequence")
technique.addItem("rmsd")

go         = QPushButton('Align', widget)
go.setToolTip('Align two molecules')
go.setEnabled(False)
```

(continues on next page)

(continued from previous page)

```
gbox = QGridLayout(widget)
gbox.addWidget( refmolName, 0,   0 )
gbox.addWidget( fitmolName, 1,   0 )
gbox.addWidget( refmolID,   0,   1 )
gbox.addWidget( fitmolID,   1,   1 )
gbox.addWidget( refmol,     0,   2 )
gbox.addWidget( fitmol,     1,   2 )
gbox.addWidget( technique,  2,   1 )
gbox.addWidget( go,         2,   2 )

widget.setLayout(gbox)
dock.setWidget(widget)
```

This next section shows the definition of all the functions that provide the interface between the widget and the desired actions.

```
def CanAlign(refID = refmolID, fitID = fitmolID, btn = go):
  try:
    id1 = int(refID.text())
    id2 = int(fitmolID.text())
    btn.setEnabled(bool(id1 and id2))
  except:
    btn.setEnabled(False)

def GetFitMol(refID = refmolID, fitID = fitmolID, btn = go):
  try:
    id = PromptID(['M'])
    if id:
        fitID.setText(str(id))
  except:
    pass

  CanAlign(refID, fitID, btn)

def GetRefMol(refID = refmolID, fitID = fitmolID, btn = go):
  try:
    id = PromptID(['M'])
    if id:
        refID.setText(str(id))
  except:
    pass

  CanAlign(refID, fitID, btn)

def Align(refID = refmolID, fitID = fitmolID, method = technique):
  try:
    try:
      id1 = int(refID.text())
      id2 = int(fitID.text())
    except:
      return
```

(continues on next page)

(continued from previous page)

```
    if id1 == 0 or id2 == 0:
      return

    if method.currentText == "shape+color":
      DoShapeAlign(id1, id2)
    elif method.currentText == "rmsd":
      DoRMSDAlign(id1, id2)
    elif method.currentText == "sequence":
      DoSequenceAlign(id1, id2)

  except:
    import traceback
    traceback.print_exc()
```

This final section shows how the widget buttons are connected up to the support functions defined in the section above.

```
QObject.connect(refmol, SIGNAL("clicked()"), GetRefMol)
QObject.connect(fitmol, SIGNAL("clicked()"), GetFitMol)
QObject.connect(go, SIGNAL("clicked()"), Align)
```

16.2.13 open_from_pdb.vpy

This example shows how to load one or molecule molecules directly from the PDB. The following section details the function which fetches the molecule(s) from the PDB corresponding to the specified PDB code(s) and then loads them into VIDA.

```
import urllib.request

def LoadMoleculesFromPDB( ids ):
  if not ids:
    return

  loaded = []

  WaitBegin()
  try:
    files = []
    for id in ids.split(","):
      id  = id.strip()
      url = urllib.request.urlopen("https://files.rcsb.org/download/" + id + ".pdb.gz")
      if url:
        AppStatusTextSet("Fetching '" + id + "i", 0)
        data = url.read()
        if len(data) > 0:
          f = open(id + ".pdb.gz", "wb")
          f.write(data)
          f.close()
          files.append(id + ".pdb.gz")
        else:
          raise IOError("PDB id (" + id + ") not found")
```

(continues on next page)

(continued from previous page)

```
    if files:
      loaded = Open(files)

  finally:
    WaitEnd()

  return loaded
```

This final section shows how a new menu item is created to prompt the user for the desired PDB codes and then to call the above function. This section also shows how to replace the command associated with a menu item in the event that the menu item has already been created.

```
menu    = "Open Special"
name    = "From PDB..."
command = "LoadMoleculesFromPDB(PromptString('Enter PDB code(s)'))"

if MenuHasItem(menu, name):
  MenuButtonActionSet(menu, name, command)
else:
  MenuAddButton(menu, name, command)
```

16.2.14 override_open_1.vpy

This example shows how to override the `Open` function in VIDA to customize the behavior when a file is opened. In this case, the original function is cached so that it can be called later within the new function. Once the file has been loaded, this function iterates over all the newly loaded molecules looking for atoms with formal charges. If a formal charge is found, a label corresponding to the charge is added to that atom.

```
_oldOpen = Open

def Open( *a, **kw ):
  PushIgnoreHint(True)
  try:
    ids   = _oldOpen(*a, **kw)
    total = len(ids)
    count = 0

    for id in ids:
      hasChg = False
      mol    = MoleculeGet(id)
      for atom in mol.GetAtoms():
        if atom.GetFormalCharge() != 0:
          hasChg = True
          break

      if hasChg:
        for key in KeysGet(id):
          AtomLabelSet(key, "%f")

      ProgressbarUpdate(count, total)
      count  = count + 1
```

(continues on next page)

(continued from previous page)

```
  finally:
    ProgressbarUpdate(0, 0)
    PopIgnoreHint()

  return ids
```

16.2.15 override_open_2.vpy

This example shows how to override the `Open` function in VIDA to customize the behavior when a file is opened. In this case, the original function is cached so that it can be called later within the new function. Once the file has been loaded, this function iterates over all the newly loaded molecules looking for molecules that do not have a dimension of 3. In cases where the dimension is not 3, 3D coordinates are generated for the molecule using Omega.

```
from openeye import oechem
from openeye import oeomega

omega = oeomega.OEOmega()
omega.SetMaxConfs(1)

_oldOpen = Open

def Open( *a, **kw ):
  global omega

  WaitBegin()
  PushIgnoreHint(True)

  # Suppress any informational and warning messages generated
  # in the following block of code

  level = oechem.OEThrow.GetLevel()
  oechem.OEThrow.SetLevel(oechem.OEErrorLevel_Error)

  try:
    ids   = _oldOpen(*a, **kw)
    total = len(ids)
    count = 0

    for id in ids:
      mol = MoleculeGet(id)
      if mol and mol.GetDimension() != 3:
        mol = oechem.OEMol()
        MoleculeCheckOut(mol, id)
        omega(mol)
        MoleculeCheckIn(mol, 0)

      ProgressbarUpdate(count, total)
      count  = count + 1

  finally:
    oechem.OEThrow.SetLevel(level)
```

(continues on next page)

(continued from previous page)

```
    ProgressbarUpdate(0, 0)
    PopIgnoreHint()
    WaitEnd()

  return ids
```

16.2.16 search_chemspider.vpy

This example shows how to search ChemSpider using the active molecule as a query. The active molecule is converted to a SMILES string which is inserted into the ChemSpider search URL.

```
from openeye import oechem

def SearchChemSpider( key ):
  if not key.IsValid():
    return

  mol = MoleculeGet(key)
  if mol:
    smiles = oechem.OECreateSmiString(mol)
    if smiles:
      AppOpenUrl("http://www.chemspider.com/Search.aspx?q=" + smiles)
```

The following section shows how the above function is bound to a button which is placed in the main application toolbar.

```
icon    = "find_32"
command = "SearchChemSpider(ActiveKey())"
tooltip = "Search ChemSpider for the active molecule"
toolbar = "Application"
name    = "ToolbarChemSpiderButton"

try:
    ToolbarRemove(toolbar, name)
except:
    pass

ToolbarAddViaNamedIcon( icon, command, tooltip, toolbar, name )
```

16.2.17 search_pubchem.vpy

This example shows how to search PubChem using the active molecule as a query. An IUPAC name is generated for the active molecule using the OpenEye **Lexichem** toolkit. The generated name (if valid) is then inserted into the PubChem search URL.

```
from openeye import oechem
from openeye import oeiupac

def SearchPubChem( key ):
  if not key.IsValid():
    return
```

(continues on next page)

(continued from previous page)

```
  mol = MoleculeGet(key)
  if mol:
    name = oeiupac.OECreateIUPACName(mol)
    if name and not "blah" in name.lower():
      AppOpenUrl('http://www.ncbi.nlm.nih.gov/sites/entrez?db=pccompound&term="' + name +
↪'"')
```

The following section shows how the above function is bound to a button which is placed in the main application toolbar.

```
icon    = "find_32"
command = "SearchPubChem(ActiveKey())"
tooltip = "Search PubChem for the active molecule"
toolbar = "Application"
name    = "ToolbarPubChemButton"

try:
    ToolbarRemove(toolbar, name)
except:
    pass

ToolbarAddViaNamedIcon( icon, command, tooltip, toolbar, name )
```

16.2.18 simple_pyside2.vpy

This example shows how to create a very simple widget with a functioning button using PySide2 and how to display it within VIDA.

```
from PySide2.QtCore    import *
from PySide2.QtWidgets import *

class SimpleWidget(QWidget):

  def __init__(self):
    QWidget.__init__(self)
    self.setWindowTitle("Example")

    self.button = QPushButton(self)
    self.button.setText("Push Me")
    self.layout = QVBoxLayout()
    self.layout.addWidget(self.button)
    self.setLayout(self.layout)
    self.resize(200, 50)

    self.connect(self.button, SIGNAL("clicked()"), self.ButtonPushed)

  def ButtonPushed(self):
    PromptMessage("Button Pushed")

widget = SimpleWidget()
widget.show()
```

16.2.19 void_volumes.vpy

This example shows how to create a function which generates and then loads into VIDA the calculated void volume between a protein and a ligand.

```
from openeye import oechem
from openeye import oegrid
from openeye import oespicoli

def CalcVoidVolumes( proteinIDOrKey, ligandIDOrKey ):
  protein = MoleculeGet(proteinIDOrKey)
  ligand  = MoleculeGet(ligandIDOrKey)

  id = 0
  if not protein or not ligand:
    return id

  try:
    PushIgnoreHint(True)
    WaitBegin()

    oechem.OEAssignBondiVdWRadii(protein)
    oechem.OEAssignBondiVdWRadii(ligand)

    grid = oegrid.OEScalarGrid()
    oespicoli.OEMakeVoidVolume(protein, ligand, grid, 0.5)

    surf = oespicoli.OESurface()
    oespicoli.OEMakeSurfaceFromGrid(surf, grid, 0.5)
    id = SurfaceAdd(surf)
    Visible(id, True)

  finally:
    WaitEnd()
    PopIgnoreHint()

  return id
```

CHAPTER

SEVENTEEN

VIDA PYTHON API

17.1 General Functions

The functions defined in this section provide a wide variety of functionality that is generally applicable to the VIDA application as a whole.

17.1.1 About

```
void About()
```

Displays information about the VIDA product.

17.1.2 AppAddCallback

```
unsigned int AppAddCallback(changeName, callbackFunction)
```

Adds a Python callback function to be called when certain changes occur within the product. The changeName parameter is a string, and can have the following values:

- "ActiveChanged"
- "DisplayChanged"
- "LockedChanged"
- "MarkedChanged"
- "ObjectsAdded"
- "ObjectsDeleted"
- "SelectedChanged"
- "VisibleChanged"
- "StateReset"

The callbackFunction parameter is the actual Python function (not a string) to be called each time the specified change happens.

The return value is an integer callback ID, which can be used to remove the callback later. See *AppRemoveCallback*.

17.1.3 AppCurrentDir

```
std::string AppCurrentDir()
```

Returns the current working directory.

17.1.4 AppCurrentDirSet

```
bool AppCurrentDirSet(const std::string &name)
```

Sets the current working directory and returns whether or not the change was successful.

17.1.5 AppDir

```
std::string AppDir()
```

Returns the path of the directory where the application is installed.

17.1.6 AppDocDir

```
std::string AppDocDir()
```

Returns the path to the directory containing all of the documentation.

17.1.7 AppExampleDir

```
std::string AppExampleDir()
```

Returns the path to the directory containing all of the examples provided with the application.

17.1.8 AppInstallDir

```
std::string AppInstallDir()
```

Returns the path to the directory where VIDA is installed.

17.1.9 AppLastDirGet

```
std::string AppLastDirGet()
```

Returns the path to the last directory that was accessed by VIDA. However, if the user has the preference "Open Current Dir" specified, this function will return an empty string.

17.1.10 AppLastDirSet

```
void AppLastDirSet(const std::string &)
```

Stores the path to the last directory that was accessed by VIDA.

17.1.11 AppLastOpenDirGet

```
std::string AppLastOpenDirGet()
```

Returns the path to the last directory from which a file was opened. However, if the user has the preference "Open Current Dir" specified, this function will return an empty string.

17.1.12 AppLastOpenDirSet

```
void AppLastOpenDirSet(const std::string &)
```

Sets the path to the last directory from which a file was opened.

17.1.13 AppLastSaveDirGet

```
std::string AppLastSaveDirGet()
```

Returns the path to the last directory to which a file was saved. However, if the user has the preference "Open Current Dir" specified, this function will return an empty string.

17.1.14 AppLastSaveDirSet

```
void AppLastSaveDirSet(const std::string &)
```

Sets the path to the last directory to which a file was saved.

17.1.15 AppLicenseFile

```
std::string AppLicenseFile()
```

Returns the path to the license file being used by VIDA.

17.1.16 AppLicenseUpdate

```
void AppLicenseUpdate(bool promptonly)
```

Searches for a new valid license file. If `promptonly` is True, VIDA will prompt the user to specify the location of the license file. If it is False, VIDA will search the expected potential locations for a new valid license file before prompting. If a valid license file is found in one of these locations, that file will be used and the user will not be prompted.

17.1.17 AppOpenUrl

```
void AppOpenUrl(const std::string &url)
```

Opens the specified URL using the most appropriate program for the specified URL. For instance, web pages will be opened with the default web browser and files will be opened with the program that that file type is associated.

17.1.18 AppRemoveCallback

```
void AppRemoveCallback(changeName, unsigned int callbackID)
```

Removes a callback created using the AppAddCallback function. The callbackID argument is the value returned from AppAddCallback, and the changeName string must be the same one that was used in the call to AppAddCallback. See *AppAddCallback*.

17.1.19 AppScriptSave

```
bool AppScriptSave(const std::string &fname)
```

Saves the current scripting history to the specified file.

17.1.20 AppUserDir

```
std::string AppUserDir()
```

Returns the path of the application user directory containing the application preferences, settings, startup script, journal file, and license.

17.1.21 AppVersion

```
std::string AppVersion()
```

Returns a string representation of the complete version number

17.1.22 AppVersionMajor

```
int AppVersionMajor()
```

Returns the first, or major, part of the version number as an integer

17.1.23 AppVersionMinor

```
int AppVersionMinor()
```

Returns the second, or minor, part of the version number as an integer

17.1.24 AppVersionBugFix

```
int AppVersionBugFix()
```

Returns the third, or bug fix, part of the version number as an integer

17.1.25 CopyData

```
void CopyData(const std::string &data)
```

Copies the specified data onto the system clipboard where it can be pasted into other applications.

17.1.26 CopyMolecules

```
void CopyMolecules(const std::vector<OEPropDB::OEKey> &keys)
```

Copies the specified molecules onto the system clipboard where they can be pasted into other applications in a variety of formats.

17.1.27 CopyMoleculesScoped

```
void CopyMoleculesScoped(unsigned int scope = BestScope)
```

Copies all of the molecules in the specified scope onto the system clipboard where they can be pasted into other applications in a variety of formats.

17.1.28 Error

```
void Error(const std::string &err)
```

Raises an exception when called from within a script, otherwise, it reports the specified error message to the status bar.

17.1.29 ErrorDetailsDialog

```
void ErrorDetailsDialog(const std::string &title, const std::string &message,
                        const std::string &error)
```

Launches an error dialog that contain detailed information about the error that occurred.

17.1.30 ExtensionsEdit

```
void ExtensionsEdit()
```

Launches the extensions manager.

17.1.31 IsLicensed

```
bool IsLicensed(const std::string &toolkit)
```

Returns whether or not a valid license exists for the specified toolkit. If a valid license is present, that toolkit can be accessed from Python in VIDA. If a valid license is not present, importing that toolkit in Python will fail and throw an exception.

17.1.32 JournalInit

```
void JournalInit(bool force)
```

For internal use only. This function specifies the script as a "Journal Script" which forces the application to clear its current state and run the script starting from a clean state.

17.1.33 ObservableBool

```
BoolObservable &ObservableBool(const std::string &name, bool create=False)
```

Returns a reference to an observable boolean value.

17.1.34 ObservableFloat

```
FloatObservable &ObservableFloat(const std::string &name, bool create=False)
```

Returns a reference to an observable floating point value.

17.1.35 ObservableInt

```
IntObservable &ObservableInt(const std::string &name, bool create=False)
```

Returns a reference to an observable integer value.

17.1.36 ObservableKey

```
KeyObservable &ObservableKey(const std::string &name, bool create=False)
```

Returns a reference to an observable OEKey object.

17.1.37 ObservableString

```
StringObservable &ObservableString(const std::string &name, bool create=False)
```

Returns a reference to an observable string value.

17.1.38 ObservableUInt

```
UIntObservable &ObservableUInt(const std::string &name, bool create=False)
```

Returns a reference to an observable unsigned integer value.

17.1.39 ObservableUpdate

```
UpdateObservable &ObservableUpdate(const std::string &name, bool create=False)
```

Returns a reference to an observable updater.

17.1.40 ObservableVecFloat

```
FloatVectorObservable &ObservableVecFloat(const std::string &name,
                                          bool create=False)
```

Returns a reference to an observable list of floating point values.

17.1.41 ObservableVecInt

```
IntVectorObservable &ObservableVecInt(const std::string &name,
                                      bool create=False)
```

Returns a reference to an observable list of integer values.

17.1.42 ObservableVecString

```
StringVectorObservable &ObservableVecString(const std::string &name,
                                            bool create=False)
```

Returns a reference to an observable list of string values.

17.1.43 ObservableVecUInt

```
UIntVectorObservable &ObservableVecUInt(const std::string &name,
                                        bool create=False)
```

Returns a reference an observable list of unsigned integer values.

17.1.44 Open

```
std::vector<unsigned int> Open(const std::string &filename,
                               const std::string &type="")
std::vector<unsigned int> Open(const std::vector<std::string> &filenames,
                               const std::string &type="")
```

This function opens the specified file or files. The optional type parameter is the file extension to assume for this file.

17.1.45 OpenState

```
void OpenState(const std::string &filename)
```

This function opens the specified state file. This action will clear the current session, including deleting all the loaded molecules.

17.1.46 PasteMolecules

```
void PasteMolecules()
void PasteMolecules(const std::vector<std::string> &,
                    const std::string &format)
```

Pastes all of the molecules that can be found in the system clipboard into the application. Pasted molecules will appear in a new list called “Pasted”.

17.1.47 PreferenceDump

```
void PreferenceDump()
```

This function prints out all the current preferences.

17.1.48 PreferenceGetBool

```
bool PreferenceGetBool(const std::string &pref, bool def=False)
```

This function returns the current boolean value of the given preference.

17.1.49 PreferenceGetColor

```
OESystem::OEColor PreferenceGetColor(const std::string &pref,
                                     const OESystem::OEColor &def=OESystem::OEWhite)
```

This function returns the current color of the given preference.

17.1.50 PreferenceGetDouble

```
double PreferenceGetDouble(const std::string &pref, double def=0.0)
```

This function returns the double precision number (non-integral) value of the given preference.

17.1.51 PreferenceGetFloat

```
float PreferenceGetFloat(const std::string &pref, float def=0.0f)
```

This function returns the floating point (non-integral) value of the given preference.

17.1.52 PreferenceGetInt

```
int PreferenceGetInt(const std::string &pref, int def=0)
```

This function returns the integer value of the given preference.

17.1.53 PreferenceGetString

```
std::string PreferenceGetString(const std::string &pref,
                                const std::string &def="")
```

This function returns the string value of the given preference.

17.1.54 PreferenceGetVBool

```
std::vector<bool> PreferenceGetVBool(const std::string &pref,
                                     const std::vector<bool> &v=std::vector<bool>())
```

This function returns the vector of boolean values for the given preference.

17.1.55 PreferenceGetVDouble

```
std::vector<double> PreferenceGetVDouble(const std::string &pref,
                                         const std::vector<double> &v=std::vector<double>
↪())
```

This function returns the vector of double precision values for the given preference.

17.1.56 PreferenceGetVFloat

```
std::vector<float> PreferenceGetVFloat(const std::string &pref,
                                       const std::vector<float> &v=std::vector<float>())
```

This function returns the vector of floating point values for the given preference.

17.1.57 PreferenceGetVInt

```
std::vector<int> PreferenceGetVInt(const std::string &pref,
                                   const std::vector<int> &v=std::vector<int>())
```

This function returns the vector of integer values for the given preference.

17.1.58 PreferenceIsBool

```
bool PreferenceIsBool(const std::string &pref)
```

This function returns whether the given preference is a boolean preference.

17.1.59 PreferenceIsColor

```
bool PreferenceIsColor(const std::string &pref)
```

This function returns whether the given preference is a color.

17.1.60 PreferenceIsDouble

```
bool PreferenceIsDouble(const std::string &pref)
```

This function returns whether the given preference is a double precision number.

17.1.61 PreferenceIsFloat

```
bool PreferenceIsFloat(const std::string &pref)
```

This function returns whether the given preference is a floating point number.

17.1.62 PreferenceIsInt

```
bool PreferenceIsInt(const std::string &pref)
```

This function returns whether the given preference is an integer.

17.1.63 PreferenceIsSet

```
bool PreferenceIsSet(const std::string &pref)
```

This function returns whether the given preference has a value.

17.1.64 PreferenceIsString

```
bool PreferenceIsString(const std::string &pref)
```

This function returns whether the given preference is a string.

17.1.65 PreferenceIsVBool

```
bool PreferenceIsVBool(const std::string &pref)
```

This function returns whether the given preference is a list of booleans.

17.1.66 PreferenceIsVDouble

```
bool PreferenceIsVDouble(const std::string &pref)
```

This function returns whether the given preference is a list of double precision numbers.

17.1.67 PreferenceIsVFloat

```
bool PreferenceIsVFloat(const std::string &pref)
```

This function returns whether the given preference is a list of floating point numbers.

17.1.68 PreferenceIsVInt

```
bool PreferenceIsVInt(const std::string &pref)
```

This function returns whether the given preference is a list of floating point numbers.

17.1.69 PreferenceMark

```
void PreferenceMark()
```

This function marks the preferences as a point for undoing.

17.1.70 PreferenceRemove

```
bool PreferenceRemove(const std::string &pref)
```

This function deletes a preference.

17.1.71 PreferenceRestore

```
void PreferenceRestore()
```

This function restores the default preferences.

17.1.72 PreferenceSetBool

```
bool PreferenceSetBool(const std::string &pref, bool v)
```

This function sets the boolean value of the preference.

17.1.73 PreferenceSetColor

```
bool PreferenceSetColor(const std::string &pref, const OESystem::OEColor &v)
```

This function sets the color value of the preference.

17.1.74 PreferenceSetCommand

```
bool PreferenceSetCommand(const std::string &pref, const std::string &cmd)
```

This function sets the command to activate the changes in the preference.

17.1.75 PreferenceSetDouble

```
bool PreferenceSetDouble(const std::string &pref, double v)
```

This function sets double precision value of the preference.

17.1.76 PreferenceSetFloat

```
bool PreferenceSetFloat(const std::string &pref, float v)
```

This function sets floating point value of the preference.

17.1.77 PreferenceSetInt

```
bool PreferenceSetInt(const std::string &pref, int v)
```

This function sets integer value of the preference.

17.1.78 PreferenceSetString

```
bool PreferenceSetString(const std::string &pref, const std::string &v)
```

This function sets the string value of the preference.

17.1.79 PreferenceSetVBool

```
bool PreferenceSetVBool(const std::string &pref, const std::vector<bool> &v)
```

This function sets value of the preference to the passed list of booleans.

17.1.80 PreferenceSetVDouble

```
bool PreferenceSetVDouble(const std::string &pref,
                          const std::vector<double> &v)
```

This function sets value of the preference to the passed list of double precision numbers.

17.1.81 PreferenceSetVFloat

```
bool PreferenceSetVFloat(const std::string &pref,
                         const std::vector<float> &v)
```

This function sets value of the preference to the passed list of floating point numbers.

17.1.82 PreferenceSetVInt

```
bool PreferenceSetVInt(const std::string &pref, const std::vector<int> &v)
```

This function sets value of the preference to the passed list of integers.

17.1.83 PreferenceValidateCommands

```
std::string PreferenceValidateCommands()
```

This function checks all the commands for the preferences to ensure that they all exist. It returns a list of Preference commands that could not be found.

17.1.84 PreferencesEdit

```
void PreferencesEdit()
```

This function opens the preferences dialog.

17.1.85 Quit

```
void Quit(bool force=False)
void quit(bool force=False)
```

Exits the application. If *force* is set to `False`, the user will be prompted before actually exiting.

17.1.86 Redo

```
void Redo()
void Redo(unsigned int steps)
```

This function redoes any previously undone actions.

17.1.87 RedoTo

```
bool RedoTo(const std::string &)
```

Redoes all the previously undone operation up to the specified location in the undo stack.

(See *UndoMark*.)

17.1.88 Save

```
void Save(const std::string &filename, const std::vector<unsigned int> &ids)
void Save(const std::string &filename,
          const std::vector<OEPropDB::OEKey> &keys)
```

This function saves the passed IDs or OEKeys to a file with the given filename.

17.1.89 SaveMiniState

```
void SaveMiniState(const std::string &filename, const std::string &version="")
```

This function saves the current application state as a "mini state" to the specified file.

17.1.90 SaveState

```
void SaveState(const std::string &filename, const std::string &version="")
```

This function saves the current application state to the specified file.

17.1.91 SaveStateFilter

```
void SaveStateFilter(const std::string &filename, const std::string &filter)
```

This function saves the state of the application to the specified file in the state file format determined by the specified *filter*.

17.1.92 SettingsGetBool

```
bool SettingsGetBool(const std::string &s, bool def=False)
```

This function returns the specified boolean setting.

17.1.93 SettingsGetFloat

```
float SettingsGetFloat(const std::string &s, float def=0.0f)
```

This function returns the specified float setting.

17.1.94 SettingsGetInt

```
int SettingsGetInt(const std::string &s, int def=0)
```

This function returns the specified int setting.

17.1.95 SettingsGetString

```
std::string SettingsGetString(const std::string &s, const std::string &def="")
```

This function returns the specified string setting.

17.1.96 SettingsRestore

```
void SettingsRestore()
```

This function returns the specified restore setting.

17.1.97 SettingsSetBool

```
bool SettingsSetBool(const std::string &s, bool v)
```

This function sets the specified bool setting.

17.1.98 SettingsSetFloat

```
bool SettingsSetFloat(const std::string &s, float v)
```

This function sets the specified float setting.

17.1.99 SettingsSetInt

```
bool SettingsSetInt(const std::string &s, int v)
```

This function sets the specified integer setting.

17.1.100 SettingsSetString

```
bool SettingsSetString(const std::string &s, const std::string &v)
```

This function sets the specified string setting.

17.1.101 SettingsSync

```
void SettingsSync()
```

This function synchronizes the application settings with those stored on disk. Ordinarily this function should not need to be called.

17.1.102 StateMark

```
int StateMark()
```

Reserved for internal use.

17.1.103 StatePop

```
int StatePop(bool load=True)
```

Reserved for internal use.

17.1.104 Undo

```
void Undo()
void Undo(unsigned int steps)
```

This function undoes the previous actions.

17.1.105 UndoHint

```
void UndoHint(const std::string &action, bool replace=True)
```

This function is used to name the current undo action. Ordinarily this function should not need to be called by the user.

17.1.106 UndoMark

```
bool UndoMark(const std::string &)
```

Sets a name for the current position in the undo stack which can be used later by the *UndoTo* function.

17.1.107 UndoTo

```
bool UndoTo(const std::string &)
```

Undoes all the previous operations up to the specified operation.

(See *UndoMark*.)

17.1.108 UpdateRedo

```
void UpdateRedo()
```

This function updates the redo menu. Ordinarily this function should not need to be called by the user.

17.1.109 VFSleep

```
void VFSleep(unsigned int seconds)
```

Sleeps the application for the specified number of seconds.

17.1.110 ViewerExportPOVRAY

```
void ViewerExportPOVRAY(const std::string &fname)
```

This function exports the current scene in the 3D display to a POV-Ray input file. The resulting file can be used as input to the POV-Ray raytracing program to generate very high quality static 3D images.

17.1.111 ViewerScreenshot

```
void ViewerScreenshot(const std::string &fname)
void ViewerScreenshot(const std::string &fname, unsigned int width,
                      unsigned int height)
```

This function saves a screenshot of the 3D display window to the specified file (*fname*) in PNG format.

17.1.112 WriteHistory

```
void WriteHistory(const std::string &filename)
```

Writes the current Python interpreter history to the specified filename.

17.2 Scope Functions

The functions detailed in this section enable the user to access information about and change the state of the application through the multiple different available scopes.

17.2.1 Active

```
void Active(const OEPropDB::OEKey &k)
void Active(unsigned int id, unsigned int conf=UINT_MAX)
```

Makes the specified object the *Active* or *Focused* object in the application. The object is specified by passing either an OEKey object or a unique ID paired with a conformer index (if dealing with multiconformer molecules).

17.2.2 ActiveConformer

```
unsigned int ActiveConformer()
```

Returns the index of the currently *Active* or *Focused* conformer.

17.2.3 ActiveID

```
unsigned int ActiveID()
```

Returns the unique ID associated with the currently *Active* or *Focused* object.

17.2.4 ActiveKey

```
OEPropDB::OEKey ActiveKey()
```

Returns the unique key associated with the currently *Active* or *Focused* object.

17.2.5 ClearActive

```
void ClearActive()
```

Clears the *active* / *focused* property from the currently *active* object.

17.2.6 ClearLocked

```
void ClearLocked()
```

Clears the locked property from all objects.

17.2.7 ClearMarked

```
void ClearMarked()
```

Clears the marked property from all objects.

17.2.8 ClearSelection

```
void ClearSelection()
```

Clears the selected property from all objects.

17.2.9 ClearVisible

```
void ClearVisible()
```

Clears the visible property from all objects.

17.2.10 ContextClear

```
void ContextClear()
```

Clears the context scope.

17.2.11 ContextGet

```
std::vector<OEPropDB::OEKey> ContextGet()
```

Returns the list of keys currently contained in the context scope. The "context" scope is used to determine which operations are appropriate for display in the right-click popup menu.

17.2.12 ContextInsert

```
void ContextInsert(const OEPropDB::OEKey &)
void ContextInsert(const std::vector<OEPropDB::OEKey> &)
```

Inserts the specified key(s) into the context scope. The "context" scope is used to determine which operations are appropriate for display in the right-click popup menu.

17.2.13 ContextKeysSet

```
void ContextKeysSet(const OEPropDB::OEKeyGroup &k)
void ContextKeysSet(const std::vector<OEPropDB::OEKey> &keys)
```

Sets the specified keys to be the "context" keys which are used to determine which operations are appropriate for display in the right-click popup menu.

17.2.14 ContextSet

```
void ContextSet(const OEPropDB::OEKey &)
void ContextSet(const std::string &prop)
void ContextSet(const std::vector<OEPropDB::OEKey> &)
```

Sets the specified key(s) to be the "context" keys which are used to determine which operations are appropriate for display in the right-click popup menu.

17.2.15 DefaultScopeGet

```
unsigned int DefaultScopeGet()
```

Returns the default scope of operations.

17.2.16 DefaultScopeSet

```
void DefaultScopeSet(unsigned int scope)
```

Sets the default scope of operations. Valid parameters are:

- **ActiveScope - operations are performed on the *active* / *focused*** object
- **ActiveThenVisibleScope - operations are performed only on the *active* /** *focused* object; however, if no object has that property, the operations are performed on the *visible* objects
- **AllScope - operations are performed on all of the currently loaded** objects
- **BestScope - allows the application to determine the current best scope** for operations based on the context of the operation
- MarkedScope - operations are performed on all of the *marked* objects
- **ScratchScope - operations are performed on all of the objects that were** added to the *scratch* scope using the *ScratchInsert* and the *ScratchSet* functions.
- SelectedScope - operations are performed only on *selected* objects
- VisibleScope - operations are performed on all of the *visible* objects

17.2.17 GetAllContextKeys

```
std::vector<OEPropDB::OEKey> GetAllContextKeys()
```

Returns all of the relevant keys of interest corresponding to the current state of the application.

17.2.18 GetAtomsScoped

```
std::vector<OEPropDB::OEKey> GetAtomsScoped(unsigned int scope)
```

Returns a list of all the atom keys in the specified scope.

17.2.19 GetBondsScoped

```
std::vector<OEPropDB::OEKey> GetBondsScoped(unsigned int scope)
```

Returns a list of all the bond keys in the specified scope.

17.2.20 GetContoursScoped

```
std::vector<OEPropDB::OEKey> GetContoursScoped(unsigned int scope)
```

Returns a list of all the contour keys in the specified scope.

17.2.21 GetGridsScoped

```
std::vector<OEPropDB::OEKey> GetGridsScoped(unsigned int scope)
```

Returns a list of all the grid keys in the specified scope.

17.2.22 GetMarkedAtoms

```
std::vector<OEPropDB::OEKey> GetMarkedAtoms()
```

Returns a list of keys corresponding to all of the currently marked atoms.

17.2.23 GetMarkedAtomsScoped

```
std::vector<OEPropDB::OEKey> GetMarkedAtomsScoped(unsigned int scope)
```

Returns a list of keys corresponding to all of the currently marked atoms that are also in the specified scope.

17.2.24 GetMarkedBonds

```
std::vector<OEPropDB::OEKey> GetMarkedBonds()
```

Returns a list of keys corresponding to all of the currently marked bonds.

17.2.25 GetMarkedBondsScoped

```
std::vector<OEPropDB::OEKey> GetMarkedBondsScoped(unsigned int scope)
```

Returns a list of keys corresponding to all of the currently marked bonds that are also in the specified scope.

17.2.26 GetMarkedContours

```
std::vector<OEPropDB::OEKey> GetMarkedContours()
```

Returns a list of keys corresponding to all of the currently marked contours.

17.2.27 GetMarkedGrids

```
std::vector<OEPropDB::OEKey> GetMarkedGrids()
```

Returns a list of keys corresponding to all of the currently marked grids.

17.2.28 GetMarkedMolecules

```
std::vector<OEPropDB::OEKey> GetMarkedMolecules()
```

Returns a list of keys corresponding to all of the currently marked molecules.

17.2.29 GetMarkedMonitors

```
std::vector<OEPropDB::OEKey> GetMarkedMonitors()
```

Returns a list of keys corresponding to all of the currently marked monitors.

17.2.30 GetMarkedSurfaces

```
std::vector<OEPropDB::OEKey> GetMarkedSurfaces()
```

Returns a list of keys corresponding to all of the currently marked surfaces.

17.2.31 GetMoleculesScoped

```
std::vector<OEPropDB::OEKey> GetMoleculesScoped(unsigned int scope)
```

Returns a list of all the molecule keys in the specified scope.

17.2.32 GetScoped

```
std::vector<OEPropDB::OEKey> GetScoped(unsigned int scope)
std::vector<OEPropDB::OEKey> GetScoped(unsigned int scope, unsigned int type)
```

Returns a list of all the keys in the specified scope. If a *type* parameter is specified, it will return a list of all keys of that type in the specified scope. The type parameter can be retrieved by calling the `GetType` method on an `OEKey` object.

17.2.33 GetSelectedAtom

```
OEPropDB::OEKey GetSelectedAtom()
```

Returns a key corresponding to a single selected atom. If multiple atoms are currently selected, this is equivalent to returning the first entry in list returned by *GetSelectedAtoms*.

17.2.34 GetSelectedAtoms

```
std::vector<OEPropDB::OEKey> GetSelectedAtoms()
```

Returns a list of keys corresponding to the currently selected atoms.

17.2.35 GetSelectedBond

```
OEPropDB::OEKey GetSelectedBond()
```

Returns a key corresponding to a single selected bond. If multiple atoms are currently selected, this is equivalent to returning the first entry in list returned by *GetSelectedBonds*.

17.2.36 GetSelectedBonds

```
std::vector<OEPropDB::OEKey> GetSelectedBonds()
```

Returns a list of keys corresponding to the currently selected bonds.

17.2.37 GetSelectedContours

```
std::vector<OEPropDB::OEKey> GetSelectedContours()
```

Returns a list of keys corresponding to the currently selected contours.

17.2.38 GetSelectedGrids

```
std::vector<OEPropDB::OEKey> GetSelectedGrids()
```

Returns a list of keys corresponding to the currently selected grids.

17.2.39 GetSelectedMolecules

```
std::vector<OEPropDB::OEKey> GetSelectedMolecules()
```

Returns a list of keys corresponding to the currently selected molecules.

17.2.40 GetSelectedMonitors

```
std::vector<OEPropDB::OEKey> GetSelectedMonitors()
```

Returns a list of keys corresponding to the currently selected monitors.

17.2.41 GetSelectedSurfaces

```
std::vector<OEPropDB::OEKey> GetSelectedSurfaces()
```

Returns a list of keys corresponding to the currently selected surfaces.

17.2.42 GetSurfacesScoped

```
std::vector<OEPropDB::OEKey> GetSurfacesScoped(unsigned int scope)
```

Returns a list of all the surface keys in the specified scope.

17.2.43 GetVisibleAtoms

```
std::vector<OEPropDB::OEKey> GetVisibleAtoms()
```

Returns a list of all the visible atom keys.

17.2.44 GetVisibleBonds

```
std::vector<OEPropDB::OEKey> GetVisibleBonds()
```

Returns a list of all the visible bond keys.

17.2.45 GetVisibleContours

```
std::vector<OEPropDB::OEKey> GetVisibleContours()
```

Returns a list of all the visible contour keys.

17.2.46 GetVisibleGrids

```
std::vector<OEPropDB::OEKey> GetVisibleGrids()
```

Returns a list of all the visible grid keys.

17.2.47 GetVisibleIDs

```
std::vector<unsigned int> GetVisibleIDs()
```

Returns a list of all the IDs for the current set of visible objects.

17.2.48 GetVisibleMolecules

```
std::vector<OEPropDB::OEKey> GetVisibleMolecules()
```

Returns a list of all the visible molecule keys.

17.2.49 GetVisibleMonitors

```
std::vector<OEPropDB::OEKey> GetVisibleMonitors()
```

Returns a list of all the visible monitor keys.

17.2.50 GetVisibleSurfaces

```
std::vector<OEPropDB::OEKey> GetVisibleSurfaces()
```

Returns a list of all the visible surface keys.

17.2.51 IDGet

```
unsigned int IDGet(const OESystem::OEBase &)
```

Returns the ID associated with the specified Python object.

17.2.52 IDTypeGet

```
std::string IDTypeGet(unsigned int)
```

Returns the name of the object type associated with the specified ID.

17.2.53 IsActive

```
bool IsActive(const OEPropDB::OEKey &k)
bool IsActive(unsigned int id, unsigned int conf=UINT_MAX)
```

Returns whether or not the specified object (key or ID with optional conformer index) is active/focused.

17.2.54 IsLocked

```
bool IsLocked(const OEPropDB::OEKey &k)
bool IsLocked(unsigned int id, unsigned int conf=UINT_MAX)
```

Returns whether or not the specified object (key or ID with optional conformer index) is locked.

17.2.55 IsMarked

```
bool IsMarked(const OEPropDB::OEKey &k)
bool IsMarked(unsigned int id, unsigned int conf=UINT_MAX)
```

Returns whether or not the specified object (key or ID with optional conformer index) is marked.

17.2.56 IsSelected

```
bool IsSelected(const OEPropDB::OEKey &k)
bool IsSelected(unsigned int id, unsigned int conf=UINT_MAX)
```

Returns whether or not the specified object (key or ID with optional conformer index) is selected.

17.2.57 IsTrulyVisible

```
bool IsTrulyVisible(const OEPropDB::OEKey &k)
bool IsTrulyVisible(unsigned int id, unsigned int conf=UINT_MAX)
```

Returns whether or not the specified object (key or ID with optional conformer index) is truly visible.

17.2.58 IsVisible

```
bool IsVisible(const OEPropDB::OEKey &k)
bool IsVisible(unsigned int id, unsigned int conf=UINT_MAX)
```

Returns whether or not the specified object (key or ID with optional conformer index) is visible.

17.2.59 Lock

```
void Lock(const std::string &k)
void Lock(const OEPropDB::OEKey &k, bool state)
void Lock(const std::vector<OEPropDB::OEKey> &keys, bool state)
void Lock(unsigned int id, bool state, unsigned int conf=UINT_MAX)
```

Sets the *locked* property on the specified object to the specified value (*state*). If a string value is passed to this function as opposed to an object specifier, the *locked* property will be set for all the objects that satisfy the specified query. For more details on the query language, see the relevant chapter in the main VIDA manual.

17.2.60 LockScoped

```
void LockScoped(bool state, unsigned int scope)
```

Sets the *locked* property to the specified value for all objects in the specified scope.

17.2.61 Mark

```
void Mark(const std::string &prop)
void Mark(const OEPropDB::OEKey &k, bool state)
void Mark(const std::vector<OEPropDB::OEKey> &keys, bool state)
void Mark(unsigned int id, bool state, unsigned int conf=UINT_MAX)
void Mark(const std::vector<OEPropDB::OEKey> &keys, bool state,
          unsigned char action)
```

Sets the *marked* property on the specified object to the specified value (*state*). If a string value is passed to this function as opposed to an object specifier, the *marked* property will be set for all the objects that satisfy the specified query. For more details on the query language, see the relevant chapter in the main VIDA manual.

17.2.62 MarkScoped

```
void MarkScoped(bool state, unsigned int scope)
```

Sets the *marked* property to the specified value for all objects in the specified scope.

17.2.63 MimicParentVisibilityGet

```
bool MimicParentVisibilityGet(const OEPropDB::OEKey &k)
```

Returns whether or not the visibility of the specified object is controlled by its parent's visibility.

17.2.64 MimicParentVisibilitySet

```
void MimicParentVisibilitySet(const OEPropDB::OEKey &k, bool mimic)
void MimicParentVisibilitySet(const std::vector<OEPropDB::OEKey> &k,
                              bool mimic)
```

Sets whether or not the visibility of the specified object(s) is controlled by each individual's parent's visibility.

17.2.65 MimicParentVisibilitySetScoped

```
void MimicParentVisibilitySetScoped(const std::string &tag, bool mimic,
                                    unsigned int scope = BestScope)
```

Sets whether or not the visibility of a certain set of objects with the specified scope is controlled by each individual's parent's visibility. The set of objects is determined by the presence or absence of the specified *tag* on the molecule.

17.2.66 ScratchClear

```
void ScratchClear()
```

This function clears the ScratchScope.

17.2.67 ScratchGet

```
std::vector<OEPropDB::OEKey> ScratchGet()
```

Returns all of the keys currently in the ScratchScope.

17.2.68 ScratchInsert

```
void ScratchInsert(const OEPropDB::OEKey &)
void ScratchInsert(const std::vector<OEPropDB::OEKey> &)
```

This function adds a key to the ScratchScope.

17.2.69 ScratchSet

```
void ScratchSet(const std::string &prop)
void ScratchSet(const OEPropDB::OEKey &)
void ScratchSet(const std::vector<OEPropDB::OEKey> &)
```

This function sets the contents of the ScratchScope.

17.2.70 Select

```
void Select(const std::string &prop)
void Select(const OEPropDB::OEKey &k, bool state)
void Select(const std::vector<OEPropDB::OEKey> &keys, bool state)
void Select(unsigned int id, bool state, unsigned int conf=UINT_MAX)
void Select(const std::vector<OEPropDB::OEKey> &keys, bool state,
            unsigned char action)
```

This function selects objects according to the selection language.

17.2.71 SelectAllMolecules

```
void SelectAllMolecules()
```

This function selects every molecule that is currently visible.

17.2.72 SelectByQuery

```
void SelectByQuery(const std::string &query, int action=0,
                   unsigned int scope = BestScope)
```

This function selects every molecule in the specified scope based on whether or not it matches the specified substructure *query*. Valid query inputs include SMILES, SMARTS, and IUPAC as well as common names. The IUPAC and common name to structure conversion is performed by the Ogham toolkit.

17.2.73 SelectInvert

```
void SelectInvert()
```

This function selects the currently visible, but not selected items, and deselects the visible and selected items.

17.2.74 SelectKeys

```
void SelectKeys(const std::vector<OEPropDB::OEKey> &atomsKeys)
```

This function selects the objects corresponding to each of the keys specified in the parameter list *keys*.

17.2.75 SelectOERID

```
void SelectOERID(unsigned int id, bool selected=true)
```

This function sets the selected property on the object corresponding to the specified *id*. The *selected* parameter specifies whether or not the object should either be selected or unselected.

17.2.76 SelectResidues

```
void SelectResidues(const std::vector<OEPropDB::OEKey> &keys, bool state)
```

Set the selected state of the residues containing the specified atoms.

17.2.77 SelectScoped

```
void SelectScoped(bool state, unsigned int scope)
```

Select or unselect (based on the *state* parameter) all the objects in the specified scope.

17.2.78 SelectWithin

```
void SelectWithin(float radius, bool selected)
```

This function selects all molecule objects, including atoms and bonds, within the specified radius of the currently selected objects.

17.2.79 SelectWithout

```
void SelectWithout(float radius, bool selected)
```

This function selects all molecule objects, including atoms and bonds, outside of the specified radii of the currently selected objects.

17.2.80 Subset

```
void Subset(const std::string &name, const std::string &sel)
```

Creates a named subset that can be then used in the Select routine.

17.2.81 Visible

```
void Visible(const std::string &prop)
void Visible(const OEPropDB::OEKey &k, bool state)
void Visible(const std::vector<OEPropDB::OEKey> &keys, bool state)
void Visible(unsigned int id, bool state, unsigned int conf=UINT_MAX)
void Visible(const std::vector<OEPropDB::OEKey> &keys, bool state,
             unsigned char action)
bool Visible(const std::vector<unsigned int> &ids, bool visible,
             unsigned int conf=UINT_MAX)
```

This function sets the keys that pass the specified criteria to be visible.

17.2.82 VisibleScoped

```
void VisibleScoped(bool state, unsigned int scope)
```

Sets the visibility of all the objects in the specified scope based on the *state* parameter.

17.2.83 VisualizeCommandScopeGet

```
unsigned int VisualizeCommandScopeGet()
```

Reserved for future expansion.

17.2.84 VisualizeCommandScopeSet

```
void VisualizeCommandScopeSet(unsigned int scope)
```

Reserved for future expansion.

17.3 User Interface Functions

The functions detailed in this section provide access to information about and enable customization of the VIDA user interface.

17.3.1 AppComponentNameGet

```
std::string AppComponentNameGet()
```

Returns the name of which VIDA component currently has the mouse focus.

17.3.2 AppComponentNameSet

```
void AppComponentNameSet(const std::string &name)
```

Stores the name of which VIDA component currently has the mouse focus. This function is called internally whenever the mouse focus changes to a new widget within the application. This is primarily intended for internal use and as such should not be called.

17.3.3 AppGeometryGet

```
std::vector<unsigned int> AppGeometryGet()
```

Returns a list of four unsigned integers corresponding to the width, height, x and y positions of the application.

17.3.4 AppGeometrySet

```
std::vector<unsigned int> AppGeometrySet( const std::vector<unsigned int> &geom )
std::vector<unsigned int> AppGeometrySet( unsigned int width,
                                          unsigned int height,
                                          unsigned int x,
                                          unsigned int y,
                                          unsigned int mainx ,
                                          unsigned int mainy )
```

Sets the width, height, x position, y position, central widget width, and central widget height of the application.

17.3.5 AppGlobalFontGet

```
std::string AppGlobalFontGet()
```

Returns the current global font used in the application.

17.3.6 AppGlobalFontSet

```
void AppGlobalFontSet(const std::string &font)
```

Sets the global font to be used in the application.

17.3.7 AppLayoutGet

```
std::string AppLayoutGet()
```

Returns a hexadecimal encoded string representation of the current layout.

17.3.8 AppLayoutSet

```
std::string AppLayoutSet(const std::string &geom)
```

Sets the current layout of the application based on the hexadecimal encoded layout specified by `geom`. Ordinarily this function should not need to be called by the user; however, in the event that it is called, the input to this function should always be the return value from a previous call to *AppLayoutGet*.

17.3.9 AppLayoutToIcon

```
std::vector<std::string> AppLayoutToIcon(unsigned int width  = 24,
                                         unsigned int height = 24 )
```

Returns a cartoon representation of the current layout as an image of size `width` and `height`. The image is returned in XPM format.

17.3.10 AppMainWindowGet

```
std::string AppMainWindowGet()
```

Returns the name of the current main window.

17.3.11 AppMainWindowSet

```
std::string AppMainWindowSet(const std::string &mainWindow)
```

Sets the current main window. The name of the desired main window is the input to this function.

17.3.12 AppMovableWindowsGet

```
bool AppMovableWindowsGet()
```

Returns whether or not the layout of the individual windows in the application can be changed by clicking and dragging the window to the desired location.

17.3.13 AppMovableWindowsSet

```
bool AppMovableWindowsSet(bool)
```

Sets whether or not the layout of the individual windows in the application can be changed by clicking and dragging the window to the desired location.

17.3.14 AppPerspectiveSet

```
std::string AppPerspectiveSet(const std::string &perspective)
```

Sets the current layout of the application based on the hexadecimal encoded layout specified by `perspective` while maintaining the current geometry (width, height, x and y positions). Ordinarily this function should not need to be called by the user; however, in the event that it is called, the input to this function should always be the return value of a previous call to *AppLayoutGet*.

17.3.15 AppRecordWindowEventsGet

```
bool AppRecordWindowEventsGet()
```

Returns whether or not window events are currently being recorded in the application history.

17.3.16 AppRecordWindowEventsSet

```
bool AppRecordWindowEventsSet(bool record)
```

Sets whether or not window events are recorded in the application history.

17.3.17 AppShowGet

```
std::string AppShowGet()
```

Returns the display status of the application. Potential return values are:

- "normal"
- "minimized"
- "maximized"
- "fullscreen"

17.3.18 AppShowMenuBarGet

```
bool AppShowMenuBarGet()
```

Returns whether or not the main menubar is currently visible.

17.3.19 AppShowMenuBarSet

```
void AppShowMenuBarSet(bool)
```

Sets whether or not the main menubar is currently visible.

17.3.20 AppShowSet

```
void AppShowSet(const std::string &style)
```

Sets the display status of the application. Valid options are:

- "normal"
- "minimized"
- "maximized"
- "fullscreen"

17.3.21 AppShowStatusBarGet

```
bool AppShowStatusBarGet()
```

Returns whether or not the main status bar is currently visible.

17.3.22 AppShowStatusBarSet

```
void AppShowStatusBarSet(bool)
```

Sets whether or not the main status bar is currently visible.

17.3.23 AppSloppyFocusGet

```
bool AppSloppyFocusGet()
```

Returns whether or not the window focus is changed by simply positioning the mouse over a window as opposed to requiring a click in the window to change focus (the default behavior).

17.3.24 AppSloppyFocusSet

```
bool AppSloppyFocusSet(bool sloppy)
```

Sets whether or not the window focus is changed by simply positioning the mouse over a window as opposed to requiring a click in the window to change focus (the default behavior).

17.3.25 AppStatusTextSet

```
void AppStatusTextSet(const std::string &text, unsigned int seconds)
```

Sets the displayed text in the status bar of the main application window to the specified text. If the *seconds* parameter is 0, this text will replace the default SMILES display in the status bar. If the *seconds* parameter is greater than 0, the text will only be displayed for the specified period of time, returning to the default SMILES display when the time has elapsed.

17.3.26 AppWindowStyleGet

```
std::string AppWindowStyleGet()
```

Returns the application window display style of the application. These styles typically correspond to specific operating system window managers.

17.3.27 AppWindowStyleSet

```
std::string AppWindowStyleSet(const std::string &windowStyle)
```

Sets the application window display style of the application. These styles typically correspond to specific operating system window managers.

17.3.28 BlockBegin

```
void BlockBegin()
```

Blocks user interaction with the GUI with the exception of the abort button. This function is used internally and should only be called by the user to restore already existing blocking behavior that may have been temporarily overridden to enable user interaction within a user specified script.

17.3.29 BlockEnd

```
void BlockEnd()
```

Restores user interaction with the GUI that was previously blocked by *BlockBegin*. This function is used internally and should only be called to enable user interaction (if needed) within a user specified script. Any script which calls this function should restore the blocking behavior by calling BlockBegin once the user interaction portion is complete. However, it is important to make sure that blocking is actually active in the first place (see *BlockGet*) before calling this function and then later calling BlockBegin.

17.3.30 BlockExemptGet

```
bool BlockExemptGet(const std::string &name)
```

Returns whether or not the GUI object specified by *name* is exempt from GUI blocking when Python commands are being executed.

17.3.31 BlockExemptSet

```
void BlockExemptSet(const std::string &name, bool value)
```

Sets whether or not the GUI object specified by *name* is exempt from GUI blocking when Python commands are being executed.

17.3.32 BlockGet

```
bool BlockGet()
```

Returns whether or not user interaction with the GUI is currently being blocked. User interaction is typically blocked while Python commands and scripts are being executed.

17.3.33 BuilderFocusOnProperties

```
void BuilderFocusOnProperties()
```

Displays the *Properties* tab in the Builder Window.

17.3.34 BuilderFocusOnSketcher

```
void BuilderFocusOnSketcher()
```

Displays the *Sketcher* tab in the Builder Window.

17.3.35 IconGetXPM

```
std::vector<std::string> IconGetXPM(const std::string &name)
```

Returns an XPM representation of the icon specified by name.

17.3.36 InterpreterClear

```
void InterpreterClear()
```

Clears the contents of the scripting window.

17.3.37 InterpreterPopUpdateGUI

```
void InterpreterPopUpdateGUI()
```

Specifies to the application that the normal interface updating behavior should occur after issued commands. This function is normally called in conjunction with *InterpreterPushUpdateGUI* to terminate the nonupdating behavior triggered by that function. It is important to note that these functions can be nested.

17.3.38 InterpreterPushUpdateGUI

```
void InterpreterPushUpdateGUI(bool update)
```

Specifies to the application whether or not the application should be updating the interface after this function is called. This is normally called to allow multiple commands to be issued in a row without causing an interface update after each individual call. A call to *InterpreterPopUpdateGUI* will resume the normal updating behavior. It is important to note that these calls can be nested.

17.3.39 InterpreterShowDebugTextGet

```
bool InterpreterShowDebugTextGet()
```

Returns whether or not debugging text will be displayed in the scripting window.

17.3.40 InterpreterShowDebugTextSet

```
void InterpreterShowDebugTextSet(bool show)
```

Sets whether or not debugging text will be displayed in the scripting window.

17.3.41 InterpreterTimerGet

```
bool InterpreterTimerGet()
```

Returns whether or not the application reports the time elapsed after every user entered command.

17.3.42 InterpreterTimerSet

```
bool InterpreterTimerSet(bool enable)
```

Sets whether or not the application reports the time elapsed after every user entered command.

17.3.43 InterpreterUpdatesGUI

```
bool InterpreterUpdatesGUI()
```

Returns whether or not VIDA is updating the interface after every command is completed.

17.3.44 LayoutCurrentGet

```
std::string LayoutCurrentGet()
```

Returns the name of the most recently applied named layout.

17.3.45 LayoutExists

```
bool LayoutExists(const std::string &name)
```

Returns whether or not the specified named layout exists.

17.3.46 LayoutGet

```
std::string LayoutGet(const std::string &name)
```

Returns a string representation of the specified name layout. The value returned by this function can be used as input to the following functions:

- *AppLayoutSet*
- *AppPerspectiveSet*

17.3.47 LayoutIcon

```
std::vector<std::string> LayoutIcon(const std::string &name)
```

Returns the icon associated with the specified named layout in XPM format.

17.3.48 LayoutLoad

```
bool LayoutLoad(const std::string &name)
```

Loads the specified named layout.

17.3.49 LayoutOrganizePrompt

```
std::string LayoutOrganizePrompt()
```

Launches the layout organization dialog.

17.3.50 LayoutOverrideOrder

```
void LayoutOverrideOrder(const std::vector<std::string> &names)
```

Sets the order to be used when displaying the named layouts in the layout button.

17.3.51 LayoutRemove

```
bool LayoutRemove(const std::string &name)
```

Removes the specified named layout.

17.3.52 LayoutRename

```
bool LayoutRename(const std::string &oldName, const std::string &newName)
```

Renames the specified named layout.

17.3.53 LayoutSaveCurrent

```
bool LayoutSaveCurrent(const std::string &name)
```

Saves the current application's layout using the specified name.

17.3.54 LayoutSet

```
void LayoutSet(const std::string &name, const std::string &data)
```

Creates a named layout using the specified name and layout data. The input for the *data* parameter can be obtained from *AppLayoutGet* command.

17.3.55 LayoutStickyGet

```
bool LayoutStickyGet()
```

17.3.56 LayoutStickySet

```
void LayoutStickySet(bool sticky)
```

17.3.57 Layouts

```
std::vector<std::string> Layouts()
```

Returns a list of the available named layouts.

17.3.58 ListWindowCollapseAll

```
void ListWindowCollapseAll()
```

Collapses all the visible nodes in the List Window to show only the top-level list entries.

17.3.59 ListWindowCollapseCurrent

```
void ListWindowCollapseCurrent()
```

Collapses the currently active list node in the List Window to a single top-level entry.

17.3.60 ListWindowExpandAll

```
void ListWindowExpandAll()
```

Expands all the top-level list nodes in the List Window to show their contents.

17.3.61 ListWindowExpandCurrent

```
void ListWindowExpandCurrent()
```

Expands the currently active list node in the List Window to show its contents.

17.3.62 ListWindowFirstList

```
void ListWindowFirstList()
```

Makes active the first item in the first list.

17.3.63 ListWindowFirstListItem

```
void ListWindowFirstListItem()
```

Makes active the first item in the current list.

17.3.64 ListWindowFocusOnOERID

```
void ListWindowFocusOnOERID(unsigned int id)
```

Changes the view of the list window to ensure that the specified ID is visible.

17.3.65 ListWindowGetCurrentPos

```
std::vector<unsigned int> ListWindowGetCurrentPos()
```

Returns the current active position with the list window. This function is primarily intended for internal use only.

17.3.66 ListWindowHideColumn

```
void ListWindowHideColumn(unsigned int column)
```

Hides the specified column from view in the list window.

17.3.67 ListWindowIsVisible

```
bool ListWindowIsVisible()
```

Returns whether or not the list window is currently visible.

17.3.68 ListWindowLastList

```
void ListWindowLastList()
```

Makes active the first item in the last list.

17.3.69 ListWindowLastListItem

```
void ListWindowLastListItem()
```

Makes active the last item in the current list.

17.3.70 ListWindowNavigateChildrenGet

```
bool ListWindowNavigateChildrenGet()
```

Returns whether or not to navigate through an object's children when browsing through the list window.

17.3.71 ListWindowNavigateChildrenSet

```
void ListWindowNavigateChildrenSet(bool nav)
```

Sets whether or not to navigate through an object's children when browsing through the list window.

17.3.72 ListWindowNavigateLeft

```
void ListWindowNavigateLeft()
```

Navigate to a new active object as if the user had pressed the left arrow key inside of the list window.

17.3.73 ListWindowNavigateRight

```
void ListWindowNavigateRight()
```

Navigate to a new active object as if the user had pressed the right arrow key inside of the list window.

17.3.74 ListWindowNextList

```
bool ListWindowNextList()
```

Makes active the first item in the next list.

17.3.75 ListWindowNextListItem

```
bool ListWindowNextListItem()
```

Makes active the next item in the current list.

17.3.76 ListWindowPrevList

```
bool ListWindowPrevList()
```

Makes active the first item in the previous list.

17.3.77 ListWindowPrevListItem

```
bool ListWindowPrevListItem()
```

Makes active the previous item in the current list.

17.3.78 ListWindowRowColoringGet

```
bool ListWindowRowColoringGet()
```

Returns whether or not the list window uses an alternating row coloring scheme.

17.3.79 ListWindowRowColoringSet

```
void ListWindowRowColoringSet(bool on)
```

Sets whether or not the list window uses an alternating row coloring scheme.

17.3.80 ListWindowSetCurrentPos

```
void ListWindowSetCurrentPos(std::vector<unsigned int> pos)
```

Sets the current active position within the list window. This is intended for internal use only.

17.3.81 ListWindowShowColumn

```
void ListWindowShowColumn(unsigned int column)
```

Shows the specified column in the list window.

17.3.82 MainWindowScreenshot

```
bool MainWindowScreenshot(const std::string &filename="")
```

Captures an image of the current main window and saves that image to the specified file.

17.3.83 MainWindowScreenshotPrompt

```
bool MainWindowScreenshotPrompt()
```

Opens a dialog which allows the user to specify multiple parameters regarding the creation of a screenshot of the main window.

17.3.84 MenuAddButton

```
std::string MenuAddButton(const std::string &menu, const std::string &button,
                          const std::string &command, bool last=false)
std::string MenuAddButton(const std::string &menu, const std::string &button,
                          const std::string &command, const std::string &hotkey,
                          bool last)
```

Adds a button to the specified menu which when pushed will execute the specified Python command. An optional `hotkey` for this menu button may be provided. The `last` argument, if specified, indicates whether or not this button should always appear at the end of the menu regardless of how many items are added to the menu after this.

Returns a unique name identifier which can be used to reference this button in other functions.

17.3.85 MenuAddRadioButton

```
std::string MenuAddRadioButton(const std::string &menu,
                               const std::string &button,
                               const std::string &command,
                               const std::string &updateCommand)
```

Adds a radio button to the specified menu which when pushed will execute the specified Python command. The `updateCommand` parameter is a Python command which is used to update the checked state of this button.

This function should only be called in between calls to *MenuBeginRadioGroup* and *MenuEndRadioGroup*.

Returns a unique name identifier which can be used to reference this button in other functions.

17.3.86 MenuAddSeparator

```
std::string MenuAddSeparator(const std::string &menu,
                             const std::string &name="",
                             bool last=false,
                             bool conditional=false)
```

Adds a simple line separator to the specified menu. The `last` parameter specifies whether to keep this separator at the end of the menu. The `conditional` parameter specifies whether or not to collapse multiple adjacent separators into a single separator.

Returns a unique name identifier which can be used to reference this separator in other functions.

17.3.87 MenuAddSubmenu

```
std::string MenuAddSubmenu( const std::string &parent,
                            const std::string &menu,
                            bool last = false )
std::string MenuAddSubmenu( const std::string &parent,
                            const std::string &menu,
                            const std::string &displayname,
                            bool last)
std::string MenuAddSubmenu( const std::string &parent,
                            const std::string &menu,
                            const std::string &displayName,
                            const std::string &update,
                            bool last)
```

Adds a submenu to the specified `parent` menu. The `menu` parameter is the desired name for this menu. The `displayName` parameter, if specified, contains the actual text to be shown when this menu is displayed. If no `displayName` parameter is specified, the `menu` parameter will be used. The `update` parameter specifies a Python command which will be called immediately before showing the menu. This command can be used to update the state of the menu or to generate the menu on the fly as desired. The `last` parameter specifies whether to keep this submenu at the end of the parent menu.

Returns a unique name identifier which can be used to reference this menu in other functions. Often this value will be the same as the value specified by the `menu` parameter, but it is not guaranteed to be the same due to potential name clashes.

17.3.88 MenuAddToggleButton

```
std::string MenuAddToggleButton(const std::string &menu,
                                const std::string &button,
                                const std::string &command,
                                const std::string &updateCommand,
                                bool last=false)
```

Adds a toggle button to the specified menu. Clicking on the button executes the Python code specified in the `command` parameter. The state of the toggle button is available to the Python command in the variable `OEInternal.MenuVal` as either 0 or 1. The `updateCommand` parameter specifies Python code which is called to determine the current state of the button. If this code returns a True value, the toggle will be set to on, if not, it will be set to off.

Returns a unique name identifier which can be used to reference this button in other functions.

17.3.89 MenuBeginRadioGroup

```
  void MenuBeginRadioGroup(const std::string &menu)

Begins a group of mutually exclusive radio buttons in the specified menu.
```

17.3.90 MenuButtonActionSet

```
void MenuButtonActionSet(const std::string &menu, const std::string &item,
                         const std::string &action)
```

Sets the Python command *action* to be executed by the specified button *item* in the specified menu *menu*.

17.3.91 MenuDisplayName

```
void MenuDisplayName(const std::string &menu, const std::string &item,
                     const std::string &display_name)
```

Sets the display name (i.e. the text actually shown for this menu item) for the specified menu item (button or submenu) in the specified menu.

17.3.92 MenuDynamicSet

```
void MenuDynamicSet(const std::string &menu, const std::string &cmd,
                    bool dynamic)
```

Sets whether or not the specified menu is dynamically updated or not. If the menu is set to be dynamically updated, the Python code contained in the cmd parameter will be executed immediately before showing the menu each time the menu is shown.

17.3.93 MenuEnableItem

```
void MenuEnableItem(const std::string &menu, const std::string &item,
                    bool enabled)
```

Sets whether or not the specified menu item is enabled.

17.3.94 MenuEnableItemCommand

```
void MenuEnableItemCommand(const std::string &menu, const std::string &item,
                           const std::string &command)
```

Sets a Python command which is run when a menu is updated to set whether the specified menu item is enabled or not.

17.3.95 MenuEndRadioGroup

```
void MenuEndRadioGroup(const std::string &menu)
```

Ends a group of mutually exclusive radio buttons in the specified menu.

17.3.96 MenuExists

```
bool MenuExists(const std::string &menu)
```

Returns whether or not the specified menu exists.

17.3.97 MenuGetAllItemsContainingName

```
std::vector<std::string> MenuGetAllItemsContainingName(const std::string &name)
```

Searches for all menu items that contain the string “name.”

17.3.98 MenuGetAllItems

```
   std::vector<std::string> MenuGetAllItems(const std::string &id)

Returns a vector of ids of all sub items of a particular menu.
If id is empty then it returns every menu item.
```

17.3.99 MenuHasItem

```
bool MenuHasItem(const std::string &menu, const std::string &item)
```

Returns whether or not the specified menu contains the specified menu item.

17.3.100 MenuItemIsAMenu

```
bool MenuItemIsAMenu ( const std::string &id )
```

Returns true if the id belongs to a menu as opposed to an action menu item.

17.3.101 MenuRemoveAll

```
void MenuRemoveAll(const std::string &menu)
```

Removes the entire contents of the specified menu.

17.3.102 MenuRemoveItem

```
void MenuRemoveItem(const std::string &item)
void MenuRemoveItem(const std::string &menu, const std::string &item)
```

Removes the specified menu item from the specified menu.

17.3.103 MenuUpdateItem

```
void MenuUpdateItem(const std::string &menu, const std::string &item)
```

Updates the specified menu item in the specified menu. Calling this function on a menu item will cause the associated item to execute any previously bound update command to determine the current state of the item.

17.3.104 Pick

```
void Pick(unsigned int action, const std::string &source,
          const std::vector<OEPropDB::OEKey> &keys, int replicate)
void Pick(unsigned int action, unsigned int startX, unsigned int startY,
          unsigned int endX, unsigned int endY, unsigned int width,
          unsigned int height)
```

Picks a small rectangular region on the 3D display specified by the start and end coordinates. This function is primarily intended for internal use.

17.3.105 PickAgain

```
void PickAgain(unsigned int action)
```

Repeats the previous *Pick* action.

17.3.106 PopIgnoreHint

```
void PopIgnoreHint()
```

This function, paired with PushIgnoreHint, minimizes application response to the functions called between them. Every PushIgnoreHint must be matched with a PopIgnoreHint and vice versa.

17.3.107 PopupAddButton

```
std::string PopupAddButton(const std::string &button,
                           const std::string &command, bool showOnce=false)
```

This function adds a clickable menu entry that runs the specified command to the popup menu. The last optional parameter specifies that this menu entry should only last one show of the popup menu.

17.3.108 PopupAddRadioButton

```
std::string PopupAddRadioButton(const std::string &button,
                                const std::string &command,
                                const std::string &updateCommand,
                                bool showOnce=false)
```

This function adds an exclusive clickable menu entry to the popup menu. The last optional parameter specifies that this menu entry should only last one show of the popup menu. It should only be called between *PopupBeginRadioGroup* and *PopupEndRadioGroup*.

17.3.109 PopupAddSeparator

```
std::string PopupAddSeparator(bool once=false, bool conditional=false)
```

This function adds a non-clickable line menu entry to the popup menu. The last optional parameter specifies that this menu entry should only last one show of the popup menu.

17.3.110 PopupAddSetupFunc

```
void PopupAddSetupFunc(const std::string &function)
```

This function specifies a worker function to be called before the popup menu is shown. It is called with no arguments. It is designed to allow context sensitive menu entries to be added to the popup menu.

17.3.111 PopupAddSubmenu

```
std::string PopupAddSubmenu(const std::string &child, bool showOnce=false)
```

This function adds a submenu to the popup menu. The last optional parameter specifies that this menu entry should only last one show of the popup menu.

17.3.112 PopupAddToggleButton

```
std::string PopupAddToggleButton(const std::string &button,
                                 const std::string &command,
                                 const std::string &updateCommand,
                                 bool showOnce=false, bool state=false)
```

This function adds a clickable menu entry that runs the specified command to the popup menu. The last optional parameter specifies that this menu entry should only last one show of the popup menu. The state of the toggle is passed as either 0 or 1 in the variable OEInternal.MenuVal. The return value of the update command is used to set the state of the toggle. Returning 1 or True from the update function sets the toggle on.

17.3.113 PopupBeginRadioGroup

```
void PopupBeginRadioGroup()
```

This function specifies that the following added radio buttons are exclusive.

17.3.114 PopupDisplayName

```
void PopupDisplayName(const std::string &item, const std::string &display_name)
```

This function modifies the display name of a popup menu entry.

17.3.115 PopupEnableItem

```
void PopupEnableItem(const std::string &item, bool enabled)
```

This function enables or disables the specified popup menu entry.

17.3.116 PopupEnableItemCommand

```
void PopupEnableItemCommand(const std::string &item,
                            const std::string &command)
```

This function sets the update command on the given entry in the popup menu.

17.3.117 PopupEndRadioGroup

```
void PopupEndRadioGroup()
```

This function ends the radio group begun by *PopupBeginRadioGroup*.

17.3.118 PopupRemoveAll

```
void PopupRemoveAll()
```

This function removes all entries from the popup menu.

17.3.119 PopupRemoveItem

```
void PopupRemoveItem(const std::string &item)
```

This function removes the specified entry from the popup menu.

17.3.120 ProgressbarUpdate

```
bool ProgressbarUpdate(int count, int total)
```

Updates the application progress bar based on the specified *count* and *total* parameters.

17.3.121 PromptAtomicNum

```
unsigned int PromptAtomicNum(const std::string &caption, unsigned int def=6)
```

Prompts the user to specify an element from a periodic table dialog.

17.3.122 PromptColor

```
OEColor PromptColor(const OEColor &deflt=OEColor())
```

Prompts the user to specify a color.

17.3.123 PromptError

```
bool PromptError(const std::string &)
```

This function prompts the user with an error message.

17.3.124 PromptFilename

```
std::string PromptFilename(const std::string &def="",
                           const std::string &mode="Open",
                           const std::string &type="all",
                           const std::string &msg="")
```

This function prompts the user for a filename. The parameters specify the default value, whether it should be an 'Open' or 'Save' dialog, and what kind of file.

17.3.125 PromptFilenames

```
std::vector<std::string>
  PromptFilenames(const std::vector<std::string> &def=std::vector<std::string>(),
                  const std::string &type="all", const std::string &msg="")
```

This function prompts the user for a collection of filenames. The parameters specify the default value, whether it should be an 'Open' or 'Save' dialog, and what kind of files.

17.3.126 PromptFloat

```
float PromptFloat(const std::string &prompt, float def=0.0f,
                  unsigned int prec=5)
```

This function prompts the user for a floating point number.

17.3.127 PromptFont

```
std::string PromptFont()
```

Prompts the user to specify a desired font.

17.3.128 PromptID

```
unsigned int PromptID(unsigned int filters=0xF0FF, bool preview=true)
unsigned int PromptID(std::vector<std::string> filterList, bool preview=true)
```

This function prompts the user for an ID from the currently loaded set.

Unsigned integer filters are an OR'ed combination of the following values.

- PromptFilter_Molecules
- PromptFilter_Surfaces
- PromptFilter_Grids
- PromptFilter_Lists
- PromptFilter_Reflections
- PromptFilter_Selected
- PromptFilter_Active
- PromptFilter_Marked
- PromptFilter_Visible
- PromptFilter_Unknown
- PromptFilter_All

String filters are just placed in a list:

- 'M' or 'm' for molecules
- 'S' or 's' for surfaces
- 'R' or 'r' for reflections
- 'G' or 'g' for grids
- 'L' or 'l' for lists
- 'O' or 'o' for Unknown objects

For example, to prompt for Molecules and Surfaces:

id = PromptID(PromptFilter_Molecules | PromptFilter_Surfaces)

or

id = PromptID(['M','S'])

17.3.129 PromptIDWriteFilters

```
std::vector<std::string> PromptIDWriteFilters(const std::string &fname)
```

This function returns the valid types that can be written to the type of file specified by the filename parameter.

17.3.130 PromptIDs

```
std::vector<unsigned int> PromptIDs(unsigned int filters=0xF0FF,
                                    bool preview=true)
std::vector<unsigned int> PromptIDs(const std::vector<std::string> &filterList,
                                    bool preview=true)
```

This function prompts the user for several IDs from the currently loaded set.

See *PromptID* for the list of available filters.

17.3.131 PromptInteger

```
int PromptInteger(const std::string &msg, int def=0)
```

This function prompts the user for an integer.

17.3.132 PromptKey

```
OEPropDB::OEKey PromptKey(unsigned int filters=0xF0FF, bool preview=true)
OEPropDB::OEKey PromptKey(const std::vector<std::string> &filterList,
                          bool preview=true)
```

This function prompts the user for a key.

See *PromptID* for the list of available filters.

17.3.133 PromptKeys

```
std::vector<OEPropDB::OEKey>
  PromptKeys(const std::vector<std::string> &filterList, bool preview=true)
std::vector<OEPropDB::OEKey> PromptKeys(unsigned int filters=0xF0FF,
                                        bool preview=true)
```

This function prompts the user for a collection of key.

See *PromptID* for the list of available filters.

17.3.134 PromptMessage

```
bool PromptMessage(const std::string &msg)
```

This function alerts the user with a message and an OK button.

17.3.135 PromptModeGet

```
std::string PromptModeGet()
```

17.3.136 PromptModeSet

```
bool PromptModeSet(const std::string &mod)
```

17.3.137 PromptFragment

```
std::string PromptMolecule(const std::string &caption)
```

Prompts the user to input a molecule fragment using the built-in molecule input dialog. Returns a hexadecimal encoded OEB string representation of the fragment. In addition to specifying a molecular fragment, the user will be required to define an attachment point, either by including a dummy atom or a selected atom or bond.

17.3.138 PromptMolecule

```
std::string PromptMolecule(const std::string &caption,
                           const std::string &def="", unsigned int options=21)
```

Prompts the user to specify a molecule using the built-in molecule input dialog. Returns a hexadecimal encoded OEB string representation of the molecule.

17.3.139 PromptMoleculeSplit

```
std::string PromptMoleculeSplit(unsigned int id)
```

Launches a dialog to assist in the splitting of the specified molecule.

17.3.140 PromptMulti

```
bool PromptMulti(const std::string &prompt, std::vector<std::string> names,
                 std::vector<std::string> types,
                 std::vector<std::string> defs,
                 std::vector<OEInterpreter::OEMultiTypeVar> &out)
```

This function prompts the user for several different types of variables and returns them in a list.

17.3.141 PromptQuery

```
std::string PromptQuery(const std::string &def="")
```

This function prompts the user for a query string for calls to *ListSubsetQuery*.

17.3.142 PromptResponseBool

```
void PromptResponseBool(const std::string &msg, bool v, bool forceInter)
```

This function pushes a boolean response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.143 PromptResponseCanceled

```
void PromptResponseCanceled(const std::string &)
```

This function pushes a cancel response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.144 PromptResponseColor

```
void PromptResponseColor(const std::string &type, const OESystem::OEColor &c,
                         bool forceInteractive)
```

This function pushes a color response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.145 PromptResponseFloat

```
void PromptResponseFloat(const std::string &type, float v, bool forceInter)
```

This function pushes a floating point response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.146 PromptResponseInt

```
void PromptResponseInt(const std::string &type, int v, bool forceInter=false)
```

This function pushes an integer response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.147 PromptResponseKey

```
void PromptResponseKey(const std::string &type, const OEPropDB::OEKey &key,
                       bool forceInteractive=false)
```

This function pushes a key response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.148 PromptResponseKeys

```
void PromptResponseKeys(const std::string &type,
                        std::vector<OEPropDB::OEKey> &keys,
                        bool forceInteractive=false)
```

This function pushes a collection of keys response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.149 PromptResponseString

```
void PromptResponseString(const std::string &type, const std::string &v,
                          bool forceInteractive=false)
```

This function pushes a string response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.150 PromptResponseUInt

```
void PromptResponseUInt(const std::string &type, unsigned int v,
                        bool forceInteractive=false)
```

This function pushes a positive integer response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.151 PromptResponseVectInt

```
void PromptResponseVectInt(const std::string &type,
                           const std::vector<int> &value,
                           bool forceInteractive=false)
```

This function pushes a list-of-integers response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.152 PromptResponseVectMulti

```
void PromptResponseVectMulti(const std::string &type,
                             const std::vector<OEInterpreter::OEMultiTypeVar> &value,
                             bool forceInteractive=false)
```

17.3.153 PromptResponseVectString

```
void PromptResponseVectString(const std::string &type,
                              const std::vector<std::string> &value,
                              bool forInteractive=false)
```

This function pushes a list-of-strings response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.154 PromptResponseVectUInt

```
void PromptResponseVectUInt(const std::string &type,
                            const std::vector<unsigned int> &value,
                            bool forceInteractive=false)
```

This function pushes a list-of-positive-integers response to an upcoming Prompt, allowing it to be skipped in scripts. Ordinarily this function should not need to be called by the user.

17.3.155 PromptSaveFilename

```
std::string PromptSaveFilename(const std::vector<unsigned int> &ids,
                               const std::string &msg="")
std::string PromptSaveFilename(const std::vector<OEPropDB::OEKey> &keys,
                               const std::string &msg="")
std::vector<std::string> PromptSaveFilename(const std::string &filters,
                                            const std::string &msg="")
```

This function prompts the user to select a file for saving. The *filters* parameter allows for specification of filter types to be shown in the file dialog. Individual filters should be separated by a pair of semicolons like this:

"OpenEye OEBinary (*.oeb);;All Files (*)"

The *msg* parameter allows for specification of the caption of the file dialog. This function returns a list where the first member is the selected filename and the second is the selected filter. If the user cancels the prompt, the returned list will be empty.

See *PromptID* for the list of available filters.

17.3.156 PromptSmiles

```
std::string PromptSmiles(const std::string &caption, const std::string &def="",
                         unsigned int options=21)
```

Prompts the user to specify a molecule using the built-in molecule input dialog. The molecule is returned as a SMILES string.

17.3.157 PromptString

```
std::string PromptString(const std::string &prompt="",
                         const std::string &def="", bool lineMode=true,
                         bool editable=true)
```

This function prompts the user for a string.

17.3.158 PromptStringListFromString

```
void PromptStringListFromString(const std::string &in,
                                std::vector<std::string> &options,
                                unsigned int &def)
```

This function prompts the user for a string.

17.3.159 PromptStringListToString

```
std::string PromptStringListToString(const std::vector<std::string> &options,
                                     unsigned int def)
std::string PromptStringListToString(const std::vector<std::string> &options,
                                     const std::string &def)
```

This function generates a list of strings from a formatted string. Ordinarily this function should not need to be called by the user.

17.3.160 PromptYesNo

```
bool PromptYesNo(const std::string &prompt, const std::string &key="")
bool PromptYesNo(const std::string &prompt, bool &check,
                 const std::string &key="")
```

This function prompts the user for a yes or no answer.

17.3.161 PromptYesNoSetDefault

```
bool PromptYesNoSetDefault(const std::string &key, bool def)
```

This function prompts the user for a yes or no answer.

17.3.162 PushIgnoreHint

```
void PushIgnoreHint(bool ignore)
```

This function delays update actions for the following commands until a PopIgnoreHint is called. PushIgnoreHint and PopIgnoreHints MUST match.

17.3.163 SetProgressbar

```
void SetProgressbar(int index, int total)
```

This function sets the progress bar to the specified value.

17.3.164 ToolbarAddSeparator

```
void ToolbarAddSeparator(const std::string &toolbar="Application")
```

Adds a separator to the specified toolbar.

17.3.165 ToolbarAddToggle

```
void ToolbarAddToggle(const std::string &onText, const std::string &offText,
                      const std::string &onCommand,
                      const std::string &offCommand, const std::string &desc,
                      const std::string &update,
                      const std::string &toolbar="Application",
                      const std::string &name="", int margin=5)
void ToolbarAddToggle(const std::vector<std::string> &onIcon,
                      const std::vector<std::string> &offIcon,
                      const std::string &onCommand,
                      const std::string &offCommand, const std::string &desc,
                      const std::string &update,
                      const std::string &toolbar="Application",
                      const std::string &name="")
```

Adds a single standard toggle button to the specified toolbar. Both on and off icons can be specified as parameters (in XPM format). Commands for both the on and off states can be specified and will be executed when the associated button enters that state.

17.3.166 ToolbarBeginRadio

```
  void ToolbarBeginRadio(const std::string &name,
                         const std::string &toolbar="Application")

Begins a named group of mutually exclusive radio buttons in the specified
toolbar.
```

17.3.167 ToolbarButtonEnableGet

```
bool ToolbarButtonEnableGet(const std::string &name,
                            const std::string &toolbar="Application")
```

Returns whether or not the specified button in the specified toolbar is currently enabled.

17.3.168 ToolbarButtonEnableSet

```
void ToolbarButtonEnableSet(const std::string &name, bool enabled,
                            const std::string &toolbar="Application")
```

Sets whether or not the specified button in the specified toolbar is currently enabled.

17.3.169 ToolbarCreate

```
void ToolbarCreate(const std::string &toolbar)
```

Creates a new toolbar with the specified name.

17.3.170 ToolbarEnabledGet

```
bool ToolbarEnabledGet(const std::string &toolbar="Application")
```

Returns whether or not the specified *toolbar* is currently enabled.

17.3.171 ToolbarEnabledSet

```
bool ToolbarEnabledSet(bool vis, const std::string &toolbar="Application")
```

Sets whether or not the specified *toolbar* is currently enabled.

17.3.172 ToolbarEndRadio

```
void ToolbarEndRadio(const std::string &name,
                     const std::string &toolbar="Application")
```

Ends the named group of mutually exclusive radio buttons in the specified toolbar.

17.3.173 ToolbarGetAll

```
std::vector<std::string> ToolbarGetAll()
```

Returns a list of all the existing toolbars in VIDA.

17.3.174 ToolbarItemCheckedGet

```
bool ToolbarItemCheckedGet(const std::string &name, const std::string &toolbar)
```

Returns whether or not the specified toolbar item in the specified toolbar is currently checked.

17.3.175 ToolbarItemCheckedSet

```
bool ToolbarItemCheckedSet(bool, const std::string &name,
                           const std::string &toolbar)
```

Sets whether or not the specified toolbar item in the specified toolbar is currently checked.

17.3.176 ToolbarItemUpdate

```
void ToolbarItemUpdate(const std::string &name, const std::string &toolbar)
```

Causes the specified toolbar item in the specified toolbar to update itself.

17.3.177 ToolbarItemVisibleGet

```
bool ToolbarItemVisibleGet(const std::string &name, const std::string &toolbar)
```

Returns whether or no the specified toolbar item in the specified toolbar is currently visible.

17.3.178 ToolbarItemVisibleSet

```
bool ToolbarItemVisibleSet(bool, const std::string &name,
                           const std::string &toolbar)
```

Sets whether or not the specified toolbar item in the specified toolbar is currently visible.

17.3.179 ToolbarRemove

```
void ToolbarRemove(const std::string &toolbar)
void ToolbarRemove(const std::string &toolbar, const std::string &name)
```

Removes the specified *toolbar* from VIDA. Please not that when a toolbar is removed, it is deleted from the application and not simply hidden.

17.3.180 ToolbarUpdate

```
void ToolbarUpdate(const std::string &toolbar)
```

Updates the specified toolbar.

17.3.181 ToolbarVisibleGet

```
bool ToolbarVisibleGet(const std::string &toolbar="Application")
```

Returns whether or not the specified toolbar is currently visible.

17.3.182 ToolbarVisibleSet

```
bool ToolbarVisibleSet(bool vis, const std::string &toolbar="Application")
```

Sets whether or not the specified toolbar is currently visible.

17.3.183 UpdateStyleWidget

```
void UpdateStyleWidget()
```

This function notifies the style widget that it should update itself due to changes that might affect its display.

17.3.184 UpdateUndo

```
void UpdateUndo()
```

This function updates the undo menu. Ordinarily this function should not need to be called by the user.

17.3.185 ViewerActiveAnnotation

```
OEGLTextBox *ViewerActiveAnnotation(int x=-1, int y=-1, int w=-1, int h=-1)
```

Create the active annotation.

17.3.186 ViewerActiveAnnotationBackgroundColorGet

```
OESystem::OEColor ViewerActiveAnnotationBackgroundColorGet()
```

Return the background color for the active annotation.

17.3.187 ViewerActiveAnnotationBackgroundColorSet

```
void ViewerActiveAnnotationBackgroundColorSet(const OESystem::OEColor &clr)
```

Set the background color for the active annotation.

17.3.188 ViewerActiveAnnotationClose

```
void ViewerActiveAnnotationClose()
```

Close the active annotation window.

17.3.189 ViewerActiveAnnotationFontGet

```
std::string ViewerActiveAnnotationFontGet()
```

Return the font used for the active annotation.

17.3.190 ViewerActiveAnnotationFontSet

```
void ViewerActiveAnnotationFontSet(const std::string &font)
```

Set the font used for the active annotation. Uses fonts as returned by the *PromptFont* scripting command.

17.3.191 ViewerActiveAnnotationForegroundColorGet

```
OESystem::OEColor ViewerActiveAnnotationForegroundColorGet()
```

Returns the foreground (text) color used in the active annotation.

17.3.192 ViewerActiveAnnotationForegroundColorSet

```
void ViewerActiveAnnotationForegroundColorSet(const OESystem::OEColor &clr)
```

Sets the foreground (text) color for the active annotation.

17.3.193 ViewerActiveAnnotationVisible

```
bool ViewerActiveAnnotationVisible()
```

Returns whether or not the active annotation is visible.

17.3.194 ViewerActiveDataFontSizeGet

```
unsigned int ViewerActiveDataFontSizeGet()
```

Returns the font size for the active data display.

17.3.195 ViewerActiveDataFontSizeSet

```
void ViewerActiveDataFontSizeSet(unsigned int sz)
```

Sets the font size for the active data display.

17.3.196 ViewerActiveDataHide

```
OEGLDataView *ViewerActiveDataHide()
```

Hides the active data display.

17.3.197 ViewerActiveDataShow

```
OEGLDataView *ViewerActiveDataShow()
OEGLDataView *ViewerActiveDataShow(int x, int y, int h, int w)
```

Returns and shows the active data display.

17.3.198 ViewerActiveDataVisible

```
bool ViewerActiveDataVisible()
```

Returns whether or not the active data display is currently visible in the 3D display.

17.3.199 ViewerBookmarkWidget

```
OEGLHList *ViewerBookmarkWidget()
```

Returns a reference to the bookmark control widget.

17.3.200 ViewerBookmarkWidgetFontSizeGet

```
unsigned int ViewerBookmarkWidgetFontSizeGet()
```

Returns the size of the font used in the bookmark control widget.

17.3.201 ViewerBookmarkWidgetFontSizeSet

```
void ViewerBookmarkWidgetFontSizeSet(unsigned int sz)
```

Sets the size of the font used in the bookmark control widget.

17.3.202 ViewerBookmarkWidgetHide

```
OEGLHList *ViewerBookmarkWidgetHide()
```

Hides the bookmark control widget from the 3D display.

17.3.203 ViewerBookmarkWidgetShow

```
OEGLHList *ViewerBookmarkWidgetShow()
```

Shows the bookmark control widget in the 3D display.

17.3.204 ViewerButtonImage

```
OEGLButton *ViewerButtonImage(const std::string &image,
                              const std::string &callback,
                              int x, int y, int w=-1, int h=-1)
```

Creates and returns a clickable button in the 3D display. The *image* parameter specifies the name of the icon to be displayed on the button. The *callback* parameter specifies the Python function to be executed when the button is clicked on. The position and dimensions of the button are specified by the parameters *x*, *y*, *w*, and *h*.

17.3.205 ViewerClick

```
void ViewerClick(int fxn, int x, int y, int gX, int gY, int w, int h)
```

This function performs an action in the display corresponding to a mouse click event in the 3D display window. Ordinarily, this function should not need to be called by the user.

17.3.206 ViewerCursorSet

```
void ViewerCursorSet(const std::string &mode)
```

This function sets the cursor style based on the *mode* parameter. Valid *mode* parameters are:

- arrow - sets the cursor to the standard arrow cursor
- back diagonal arrow - sets the cursor to a backwards diagonal double-ended arrow
- blank - sets the cursor to a blank cursor (no cursor will appear)
- busy - sets the cursor to be a busy indicator (an hourglass on many systems)
- closed hand - sets the cursor to be a closed hand

- cross - sets the cursor to two crossed lines
- double arrow - sets the cursor to two crossed double-ended arrows
- forbidden - sets the cursor to be a circle with a cross line through it
- forward diagonal arrow - sets the cursor to a forwards diagonal double-ended arrow
- horizontal arrow - sets the cursor to a horizontal double-ended arrow
- ibeam - sets the cursor to an ibeam style cursor
- open hand - sets the cursor to be an open hand
- pointing - sets the cursor to be a pointing hand
- rotation - same as *cross*
- split horizontal - sets the cursor to two parallel horizontal lines with perpendicular arrows coming out of each line
- split vertical - sets the cursor to two parallel vertical lines with perpendicular arrows coming out of each line
- translation - same as *double arrow*
- up arrow - sets the cursor to an upwards point arrow
- vertical arrow - sets the cursor to a vertical double-ended arrow
- wait - sets the cursor to a waiting indicator (an hourglass on many systems)
- whats this - sets the cursor to be an arrow with an adjacent question mark

17.3.207 ViewerDepict

```
OEGLImage *ViewerDepict(const OEPropDB::OEKey &key, int x, int y, int width,
                        int height)
```

Creates and returns a depiction widget for the 3D display. The *key* parameter specifies which molecule to use for the depiction. The position and dimensions of the widget are specified by the *x*, *y*, *width*, and *height* parameters.

17.3.208 ViewerDepictionAntiAliasGet

```
bool ViewerDepictionAntiAliasGet()
```

Returns whether or not the lines used in drawing depictions in the 3D display should be anti-aliased.

17.3.209 ViewerDepictionAntiAliasSet

```
void ViewerDepictionAntiAliasSet(bool s)
```

Sets whether or not the lines used in drawing depictions in the 3D display should be anti-aliased.

17.3.210 ViewerDepictionHeightGet

```
float ViewerDepictionHeightGet()
```

Returns the percent of the total height of the 3D display that the active depiction display uses when drawing onto the 3D display.

17.3.211 ViewerDepictionLineWidthGet

```
float ViewerDepictionLineWidthGet()
```

Returns the width of the lines used in drawing depictions in the 3D display.

17.3.212 ViewerDepictionLineWidthSet

```
void ViewerDepictionLineWidthSet(float lw)
```

Sets the width of the lines used in drawing depictions in the 3D display.

17.3.213 ViewerDepictionSizeSet

```
float ViewerDepictionSizeSet(float percent)
float ViewerDepictionSizeSet(float perc_w, float perc_h)
```

Sets the percents of the total width and height of the 3D display that the active depiction display uses when drawing onto the 3D display. The *perc_w* parameter specifies the percent width and the *perc_h* specifies the percent height.

17.3.214 ViewerDepictionWidthGet

```
float ViewerDepictionWidthGet()
```

Returns the percent of the total width of the 3D display that the active depiction display uses when drawing onto the 3D display.

17.3.215 ViewerLabel

```
OEGLLabel *ViewerLabel(const std::string &message, int x, int y)
```

Creates and returns a text label which is displayed in the 3D window. The text is specified by the *message* parameter and the position is specified by the *x* and *y* parameters.

17.3.216 ViewerLabelDialog

```
void ViewerLabelDialog(const std::string &type)
```

Opens a dialog which allows the user to specify what types of labels to apply to either atoms or bonds.

17.3.217 ViewerMouseClick

```
void ViewerMouseClick(unsigned int state, unsigned int action)
```

This function specifies the desired *action* that should occur when a mouse click event is generated with the associated *state*. The *state* parameter specifies which mouse buttons were clicked as well as whether any modifier keys were depressed at the same time. *State* parameters can be added together to specify complicated states. Valid *state* parameters are:

- MouseState_None
- MouseState_Left
- MouseState_Middle

- MouseState_Right
- MouseState_Shift
- MouseState_Control
- MouseState_Alt

Action parameters are distinct and cannot be added together like *state* parameters. Valid *action* parameters are:

- MouseClick_None
- MouseClick_Menu - creates a context specific popup menu
- **MouseClick_Select - selects the object that was clicked on and clears the** previous selection
- **MouseClick_SelectAdd - selects the object that was clicked on, but does not** clear the previous selection
- **MouseClick_SelectGrow - extends the selection of the object that was** clicked on: atom -> residue -> chain -> mol and clears the previous selection
- **MouseClick_SelectAddGrow - extends the selection of the object that was** clicked on: atom -> residue -> chain -> mol and clears the previous selection
- **MouseClick_SelectToggle - toggles the selection of the object that was** clicked on
- **MouseClick_Label - displays a context specific label corresponding to the** object that was clicked on

17.3.218 ViewerMouseDoubleClick

```
void ViewerMouseDoubleClick(unsigned int state, unsigned int action)
```

This function specifies the desired *action* that should occur when a mouse double-click event is generated with the associated *state*. The *state* parameter specifies which mouse buttons were clicked as well as whether any modifier keys were depressed at the same time. *State* parameters can be added together to specify complicated states. Valid *state* parameters are:

- MouseState_None
- MouseState_Left
- MouseState_Middle
- MouseState_Right
- MouseState_Shift
- MouseState_Control
- MouseState_Alt

Action parameters are distinct and cannot be added together like *state* parameters. Valid *action* parameters are:

- MouseClick_None
- MouseClick_Menu - creates a context specific popup menu
- **MouseClick_Select - selects the object that was clicked on and clears the** previous selection
- **MouseClick_SelectAdd - selects the object that was clicked on, but does not** clear the previous selection

- **MouseClick_SelectGrow - extends the selection of the object that was**
  clicked on: atom -> residue -> chain -> mol and clears the previous selection
- **MouseClick_SelectAddGrow - extends the selection of the object that was**
  clicked on: atom -> residue -> chain -> mol and clears the previous selection
- **MouseClick_SelectToggle - toggles the selection of the object that was**
  clicked on
- **MouseClick_Label - displays a context specific label corresponding to the**
  object that was clicked on

17.3.219 ViewerMouseFunctionGet

```
unsigned int ViewerMouseFunctionGet()
```

Returns the current function of the mouse when used in the 3D display. For more details on the different types of functions, please see the documentation for ViewerMouseFunctionSet.

17.3.220 ViewerMouseFunctionNameGet

```
std::string ViewerMouseFunctionNameGet(unsigned int)
```

Returns the name of the specified mouse function.

17.3.221 ViewerMouseFunctionSet

```
void ViewerMouseFunctionSet(unsigned int action)
void ViewerMouseFunctionSet(const std::string &)
```

Sets the current function of the mouse when used in the 3D display. Valid functions are:

- MouseClick_None - this tells the mouse to act according to behaviors specified by the current mouse map (this is the default behavior)
- MouseClick_Label - this mode causes a detailed information label to appear on the screen regarding whatever is currently under the cursor
- MouseClick_MeasureAngle - this puts the mouse into angle measurement mode
- MouseClick_MeasureDistance - this puts the mouse into distance measurement mode
- MouseClick_MeasureTorsion - this puts the mouse into torsion measurement mode

17.3.222 ViewerMouseMap

```
void ViewerMouseMap(const std::string &map)
```

This function sets the current mouse mapping. Valid parameters include:

- vida
- afitt
- coot
- insight
- moe

- o
- quanta
- rasmol
- sybyl
- maestro

17.3.223 ViewerMouseMove

```
void ViewerMouseMove(unsigned int state, unsigned int action,
                     unsigned int loc=OEGUI::MouseLocation::Both,
                     unsigned int dir=OEGUI::MouseDirection::Both)
```

This function specifies the desired *action* that should occur when a mouse movement event is generated with the associated *state*, in the specified *location*, and in the specified *direction*. The *state* parameter specifies which mouse buttons were clicked as well as whether any modifier keys were depressed at the same time. *State* parameters can be added together to specify complicated states. Valid *state* parameters are:

- MouseState_None
- MouseState_Left
- MouseState_Middle
- MouseState_Right
- MouseState_Shift
- MouseState_Control
- MouseState_Alt

The *location* parameter specifies whether the mouse was inside or outside a central circular region in the 3D display. The bounds of this region can be displayed using the ViewerShowTrackballGuideSet scripting command. Valid *location* parameters are:

- MouseLocation_None
- MouseLocation_Inside
- MouseLocation_Outside
- MouseLocation_Both

The *direction* parameter specifies which direction of mouse motion should control the action. Valid *direction* parameters are:

- MouseDirection_None
- MouseDirection_X
- MouseDirection_Y
- MouseDirection_Both

Action parameters are distinct and cannot be added together like *state* parameters. Valid *action* parameters are:

- MouseMotion_None
- MouseMotion_RotateX - rotates the display along the X axis
- MouseMotion_RotateY - rotates the display along the Y axis
- MouseMotion_RotateZ - rotates the display along the Z axis

- MouseMotion_Trackball - rotates the display using a trackball motion
- MouseMotion_TranslateXY - translates the display along the X and Y axes
- MouseMotion_TranslateX - translates the display along the X axis
- MouseMotion_TranslateY - translates the display along the Y axis
- MouseMotion_TranslateZ - translates the display along the Z axis
- MouseMotion_Zoom - zooms the display
- **MouseMotion_Clip - controls the position of the far and near clipping** planes in a mirrored fashion
- MouseMotion_ClipFar - controls the position of the far clipping plane
- MouseMotion_ClipNear - controls the position of the near clipping plane
- **MouseMotion_DepthcueBegin - controls the position of the start of the** depthcueing calculation
- **MouseMotion_DepthcueEnd - controls the position of the end of the** depthcueing calculation
- **MouseMotion_SelectRect - draws a rectangle on the screen and will select** the entire contents of that rectangle when the mouse button is released while clearing the previous selection
- **MouseMotion_SelectRectAdd - draws a rectangle on the screen and will select** the entire contents of that rectangle when the mouse button is released while keeping the previous selection
- **MouseMotion_Label - displays a context specific label containing** information about whatever object is currently beneath the mouse
- **MouseMotion_Contour - adjusts the contour level of the grid contours in the** current scope
- **MouseMotion_TextScale - adjusts the size of the text displayed in the 3D** display

17.3.224 ViewerMouseOutsideAwareGet

```
bool ViewerMouseOutsideAwareGet()
```

Returns whether or not the display window recognizes mouse motion along the outside edge of the window as a potentially different action from mouse motion in the interior of the window. Ordinarily, this function should not need to be called by the user.

17.3.225 ViewerMouseOutsideAwareSet

```
bool ViewerMouseOutsideAwareSet(bool aware)
```

Sets whether or not the display window recognizes mouse motion along the outside edge of the window as a potentially different action from mouse motion in the interior of the window. Ordinarily, this function should not need to be called by the user.

17.3.226 ViewerMouseReset

```
void ViewerMouseReset()
```

Clears the current mouse mapping. Ordinarily, this function should not need to be called by the user. Furthermore, calling this function without rebuilding the mouse map will disable all interaction with the 3D window.

17.3.227 ViewerMouseSensitivityGet

```
float ViewerMouseSensitivityGet()
```

Returns the sensitivity of mouse event detection in the 3D window.

17.3.228 ViewerMouseSensitivitySet

```
float ViewerMouseSensitivitySet(float s)
```

Sets the sensitivity of mouse event detection in the 3D window.

17.3.229 ViewerMouseWheel

```
void ViewerMouseWheel(unsigned int state, unsigned int action)
```

This function specifies the desired *action* that should occur when a mouse wheel event is generated with the associated *state*. The *state* parameter specifies whether any modifier keys were depressed at the same time. *State* parameters can be added together to specify complicated states. Valid *state* parameters are:

- MouseState_None
- MouseState_Shift
- MouseState_Control
- MouseState_Alt

Action parameters are distinct and cannot be added together like *state* parameters. Valid *action* parameters are:

- MouseMotion_None
- MouseMotion_RotateX - rotates the display along the X axis
- MouseMotion_RotateY - rotates the display along the Y axis
- MouseMotion_RotateZ - rotates the display along the Z axis
- MouseMotion_TranslateX - translates the display along the X axis
- MouseMotion_TranslateY - translates the display along the Y axis
- MouseMotion_TranslateZ - translates the display along the Z axis
- MouseMotion_Zoom - zooms the display
- **MouseMotion_Clip - controls the position of the far and near clipping** planes in a mirrored fashion
- MouseMotion_ClipFar - controls the position of the far clipping plane
- MouseMotion_ClipNear - controls the position of the near clipping plane
- **MouseMotion_DepthcueBegin - controls the position of the start of the** depthcueing calculation
- **MouseMotion_DepthcueEnd - controls the position of the end of the** depthcueing calculation

- **MouseMotion_Contour - adjusts the contour level of the grid contours in the** current scope
- **MouseMotion_TextScale - adjusts the size of the text displayed in the 3D** display

17.3.230 ViewerMove

```
void ViewerMove(int fxnX, int fxnY, int x, int y, int oldx, int oldy, int w, int h,
                bool first)
```

This function performs an action in the display window corresponding to a mouse move event in the 3D display window. Ordinarily, this function should not need to be called by the user.

17.3.231 ViewerPickSurfaceTrianglesGet

```
bool ViewerPickSurfaceTrianglesGet()
```

Returns whether or not picking on a surface picks individual triangles as opposed to picking the entire surface. Ordinarily, this function should not need to be called by the user.

17.3.232 ViewerPickSurfaceTrianglesSet

```
bool ViewerPickSurfaceTrianglesSet(bool enable)
```

Sets whether or not picking on a surface picks individual triangles as opposed to picking the entire surface. Ordinarily, this function should not need to be called by the user.

17.3.233 ViewerProgress

```
OEGLProgress *ViewerProgress(const std::string &message)
```

Creates and returns a progress bar display shown in the 3D window. The *message* parameter specifies the text associated with the progress bar.

17.3.234 ViewerRemoveWidget

```
void ViewerRemoveWidget(OEGLWidget *widget, bool update)
```

This function removes the specified widget from the 3D display. The *update* parameter specifies whether or not all the other widgets in the 3D display need to be updated as a result of this change.

17.3.235 ViewerTextBox

```
OEGLTextBox *ViewerTextBox(const OEPropDB::OEKey &key, int x=-1, int y=-1,
                           int w=-1, int h=-1)
OEGLTextBox *ViewerTextBox(const std::string &message, int x=-1, int y=-1,
                           int w=-1, int h=-1)
```

Creates and returns a text display widget to be shown in the 3D display. The parameter *message* specifies the current text. The position and dimensions of the widget are specified by the parameters *x*, *y*, *w*, and *h*.

17.3.236 ViewerUseInertiaGet

```
bool ViewerUseInertiaGet()
```

Returns whether or not the 3D display window tracks the inertia of mouse movements and uses that to keep the scene spinning after the mouse button is released.

17.3.237 ViewerUseInertiaSet

```
bool ViewerUseInertiaSet(bool state)
```

Sets whether or not the 3D display window tracks the inertia of mouse movements and uses that to keep the scene spinning after the mouse button is released.

17.3.238 ViewerUseKeyMapGet

```
bool ViewerUseKeyMapGet()
```

Returns whether or not the viewer is using its own internal key bindings. Ordinarily, this function does not need to be called by the user.

17.3.239 ViewerUseKeyMapSet

```
bool ViewerUseKeyMapSet(bool state)
```

Sets whether or not the viewer is using its own internal key bindings. Ordinarily, this function does not need to be called by the user.

17.3.240 ViewerWheel

```
void ViewerWheel(int fxn, int x, int y, int delta)
```

This function performs an action in the display window corresponding to a mouse wheel event in the 3D display window. Ordinarily, this function should not need to be called by the user.

17.3.241 ViewerWidgetUpdate

```
void ViewerWidgetUpdate()
```

Forces an update of all the display widgets in the 3D display.

17.3.242 WaitBegin

```
void WaitBegin()
```

This function sets the application in a waiting state. It is used for long operations. Ordinarily this function should not need to be called by the user.

17.3.243 WaitEnd

```
void WaitEnd()
```

This function stops the application in a waiting state. It is used for long operations. Ordinarily this function should not need to be called by the user.

17.3.244 WindowEnabledGet

```
bool WindowEnabledGet(const std::string &windowName)
```

Returns whether or not the specified window is enabled for use within VIDA.

17.3.245 WindowEnabledSet

```
void WindowEnabledSet(const std::string &windowName, bool enabled)
```

Sets whether or not the specified window is enabled for use within VIDA.

17.3.246 WindowMenuUpdate

```
void WindowMenuUpdate()
```

Updates the contents of the main *Window* menu.

17.3.247 WindowRegisterHotkey

```
void WindowRegisterHotkey(const std::string &windowName,
                          const std::string &keystroke,
                          const std::string &pyCommand)
```

Registers the specified keystroke with the associated Python command in the specified window. If this keystroke is detected when the specified window has the application focus, the associated command will be run.

17.3.248 WindowVisibleGet

```
bool WindowVisibleGet(const std::string &windowName)
```

Returns whether or not the specified window is currently visible.

17.3.249 WindowVisibleSet

```
bool WindowVisibleSet(const std::string &windowName, bool visible,
                      const std::string &pos="")
```

Sets whether or not the specified window is currently visible.

17.4 Display Functions

The functions detailed in this section enable the customization of how the application itself as well as the currently loaded objects are displayed.

17.4.1 AnnotationGet

```
std::string AnnotationGet(const OEPropDB::OEKey &key)
```

Returns the text annotation associated with the object specified by `key`.

17.4.2 AnnotationHas

```
bool AnnotationHas(const OEPropDB::OEKey &key)
```

Returns whether or not the object specified by `key` has a text annotation.

17.4.3 AnnotationSet

```
void AnnotationSet(const OEPropDB::OEKey &key, const std::string &txt,
                   bool notify=true)
```

Sets a text annotation on the object specified by `key` containing the text specified by `txt`. If `notify` is set to `True`, VIDA will be notified to update the user interface to reflect this change.

17.4.4 AtomClearColorScoped

```
void AtomClearColorScoped(unsigned int scope = BestScope)
```

This function restores the default atom coloring to those atoms within the specified scope.

17.4.5 AtomColorPaletteUpdate

```
void AtomColorPaletteUpdate()
```

This function updates the default atom coloring palette based on whether or not the application is using a dark or light colored background scheme. This function is called internally and should not be called by the user.

17.4.6 AtomColorReferenceScoped

```
void AtomColorReferenceScoped(unsigned int scope = BestScope)
```

This function provides a quick way to mark a molecule (or set of atoms) as a reference structure. The coloring function that is applied colors all carbon atoms the current application 'reference' color (default is green). The 'reference' color can be changed in the preferences.

17.4.7 AtomColorResidueScoped

```
void AtomColorResidueScoped(unsigned int scope = BestScope)
```

This function provides a quick way to color a molecule (or set of atoms) based on associated residue information. The default coloring that is applied uses the 'Shapely' color scheme for coloring residues. The residue colors can be changed in the preferences:

```
* ALA     Medium Green  [140,255,140]
* GLY     White         [255,255,255]
* LEU     Olive Green   [ 69, 94, 69]
* SER     Medium Orange [255,112, 66]
* VAL     Light Purple  [255,140,255]
* THR     Dark Orange   [184, 76,  0]
* LYS     Royal Blue    [ 71, 71,184]
* ASP     Dark Rose     [160,  0, 66]
* ILE     Dark Green    [  0, 76,  0]
* ASN     Light Salmon  [255,124,112]
```

(continues on next page)

(continued from previous page)

```
* GLU     Dark Brown    [102,  0,  0]
* PRO     Dark Grey     [ 82, 82, 82]
* ARG     Dark Blue     [  0,  0,124]
* PHE     Olive Grey    [ 83, 76, 66]
* GLN     Dark Salmon   [255, 76, 76]
* TYR     Medium Brown  [140,112, 76]
* HIS     Medium Blue   [112,112,255]
* CYS     Medium Yellow [255,255,112]
* MET     Light Brown   [184,160, 66]
* TRP     Olive Brown   [ 79, 70,  0]
* ASX     Medium Purple [255,  0,255]
* GLX     Medium Purple [255,  0,255]
* PCA     Medium Purple [255,  0,255]
* HYP     Medium Purple [255,  0,255]
* A       Light Blue    [160,160,255]
* C       Light Orange  [255,140, 75]
* G       Medium Salmon [255,112,112]
* T       Light Green   [160,255,160]
* Default Medium Purple [255,  0,255]
```

17.4.8 AtomColorSetScoped

```
void AtomColorSetScoped(const OESystem::OEColor &color,
                        unsigned int scope = BestScope)
void AtomColorSetScoped(unsigned int element, const OESystem::OEColor &color,
                        unsigned int scope = BestScope)
```

This function sets the color of the atoms in the specified scope to be the specified color.

17.4.9 AtomDarkColorsGet

```
bool AtomDarkColorsGet()
```

Returns whether or not the application is using the atom color palette designed for dark colored backgrounds. Otherwise, the application is using the atom color palette designed for light colored backgrounds. Both of these palettes can be edited in the preferences.

17.4.10 AtomDarkColorsSet

```
void AtomDarkColorsSet(bool dark)
```

Sets whether or not the application is using the atom color palette designed for dark colored backgrounds. Otherwise, the application is using the atom color palette designed for light colored backgrounds. Both of these palettes can be edited in the preferences.

17.4.11 AtomDefaultColorGet

```
OESystem::OEColor AtomDefaultColorGet()
OESystem::OEColor AtomDefaultColorGet(unsigned int elem)
```

Returns the current default atom color. The default atom color is applied to every atom that does not have its own specific default color assigned. Calling this function with the *elem* parameter returns the specific default color associated with

the passed element number (e.g. carbon = 6).

17.4.12 AtomDefaultColorSet

```
void AtomDefaultColorSet(const OESystem::OEColor &color)
void AtomDefaultColorSet(unsigned int elem, const OESystem::OEColor &color)
```

Sets the current default atom color. The default atom color is applied to every atom that does not have its own specific default color assigned. Calling this function with the *elem* parameter sets the specific default color associated with the passed element number (e.g. carbon = 6).

17.4.13 AtomHaloRadiusGet

```
float AtomHaloRadiusGet()
```

Returns the current radius of atom selection halo. The atom selection halo is a transparent white surface with the specified radius drawn around the current selected set of atoms. If the DistanceControlsVisibility property is set (see DistanceControlsVisibilitySet/Get functions), only atoms falling within this radius will be displayed.

17.4.14 AtomHaloRadiusSet

```
float AtomHaloRadiusSet(float radius, bool notify=true)
```

Sets the current radius of atom selection halo. The atom selection halo is a transparent white surface with the specified radius drawn around the current selected set of atoms. If the DistanceControlsVisibility property is set (see DistanceControlsVisibilitySet/Get functions), only atoms falling within this radius will be displayed.

17.4.15 AtomHydrogenStyleGet

```
unsigned int AtomHydrogenStyleGet()
unsigned int AtomHydrogenStyleGet(const OEPropDB::OEKey &key)
```

Returns the current default hydrogen display style - 0 for no hydrogens, 1 for polar hydrogens only, and 2 for all hydrogens.

17.4.16 AtomHydrogenStyleSet

```
void AtomHydrogenStyleSet(unsigned int style)
```

Sets the current default hydrogen display style. Valid *style* parameter values are 0 for no hydrogens, 1 for polar hydrogens only, and 2 for all hydrogens.

17.4.17 AtomHydrogenStyleSetScoped

```
void AtomHydrogenStyleSetScoped(unsigned int style,
                                unsigned int scope = BestScope)
```

Sets the hydrogen display style for all the molecules in the specified scope. Valid *style* parameter values are 0 for no hydrogens, 1 for polar hydrogens only, and 2 for all hydrogens.

17.4.18 AtomLabelDefaultGet

```
std::string AtomLabelDefaultGet()
```

Returns the default label displayed on atoms. The default is "None". See the AtomLabelSetScoped function for more information about label types.

17.4.19 AtomLabelDefaultSet

```
void AtomLabelDefaultSet(const std::string &label)
```

Sets the default label displayed on atoms. The default is "None". See the AtomLabelSetScoped function for more information about label types.

17.4.20 AtomLabelGet

```
std::string AtomLabelGet(const OEPropDB::OEKey &k)
```

Returns the current label set on the specified atom.

17.4.21 AtomLabelSet

```
bool AtomLabelSet(const OEPropDB::OEKey &k, const std::string &label)
```

Sets the label on the specified atom. The parameter *label* contains the text of the label and the parameter *key* specifies which atom this label should apply to. See the AtomLabelSetScoped function for more information about label types.

17.4.22 AtomLabelSetScoped

```
bool AtomLabelSetScoped(const std::string &label,
                        unsigned int scope = BestScope)
```

Sets the current label displayed on the atoms in the specified scope. Label types include:

```
* "none"               or ""
* "index"              or "%i"
* "element"            or "%e"
* "name"               or "%n"
* "type"               or "%t"
* "inttype"            or "%it"
* "implicit hcount"    or "%h"
* "formal charge"      or "%f"
* "partial charge"     or "%p"
* "residue info"       or "%r"
* "degree"             or "%d"
* "ischiral"           or "%c"
* "chirality"          or "%chi"
* "hasstereospecified" or "%sts"
* "symmetry class"     or "%sy"
* "isinring"           or "%rng"
* "oechem idx"         or "%oei"
* "map index"          or "%m"
* "read order"         or "%ro"
```

(continues on next page)

(continued from previous page)

```
* "isotope"            or "%iso"
* "hcount"             or "%h"
* "residue info (ca)"  or "%car"
* "bfactor"            or "%cab"
* "degree"             or "%d"
* "hybrid"             or "%hyb"
* "comment"            or "%cm"
* "%gd(x)" - where x specifies the desired piece of generic data to use as the label
```

17.4.23 AtomStyleGet

```
std::string AtomStyleGet()
```

Returns the default atom style. The default is "small". See the AtomStyleSetScoped function for more information about atom styles.

17.4.24 AtomStyleGetScoped

```
unsigned int AtomStyleGetScoped(unsigned int scope = BestScope)
```

Returns the style of the atoms in the specified scope. If there are multiple styles present, the return type is "mixed". See the AtomStyleSetScoped function for more information about atom styles.

17.4.25 AtomStyleLargeGet

```
std::string AtomStyleLargeGet()
```

Returns the default atom style for large molecules, where large is determined by whether or not the number of atoms in the molecule exceeds the specified cutoff. For more information on this cutoff, see the MoleculeSizeCutoff function. The default is "none". See the AtomStyleSetScoped function for more information about atom styles.

17.4.26 AtomStyleLargeSet

```
void AtomStyleLargeSet(unsigned int style)
void AtomStyleLargeSet(const std::string &atomStyle)
```

Sets the default atom style for large molecules, where large is determined by whether or not the number of atoms in the molecule exceeds the specified cutoff. This function accepts either a string or an enumerated representation as its parameter. The default is "none". For more information on this large molecule atom cutoff, see the MoleculeSizeCutoff function. See the AtomStyleSetScoped function for more information about atom styles.

17.4.27 AtomStyleNucleicGet

```
std::string AtomStyleNucleicGet()
```

Returns the default atom style for nucleic acids. The default is to mirror the large molecule atom style. See the AtomStyleLargeSet function for more information about large molecules and see the AtomStyleSetScoped function for more information about atom styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.28 AtomStyleNucleicSet

```
void AtomStyleNucleicSet(unsigned int style)
void AtomStyleNucleicSet(const std::string &atomStyle)
```

Sets the default atom style for nucleic acids. This function accepts either a string or an enumerated representation as its parameter. The default is to mirror the large molecule atom style. See the AtomStyleLargeSet function for more information about large molecules and see the AtomStyleSetScoped function for more information about atom styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.29 AtomStyleProteinGet

```
std::string AtomStyleProteinGet()
```

Returns the default atom style for proteins. The default is to mirror the large molecule atom style. See the AtomStyleLargeSet function for more information about large molecules and see the AtomStyleSetScoped function for more information about atom styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.30 AtomStyleProteinSet

```
void AtomStyleProteinSet(unsigned int style)
void AtomStyleProteinSet(const std::string &atomStyle)
```

Sets the default atom style for proteins. The default is to mirror the large molecule atom style. See the AtomStyleLargeSet function for more information about large molecules and see the AtomStyleSetScoped function for more information about atom styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.31 AtomStyleSet

```
void AtomStyleSet(unsigned int style)
void AtomStyleSet(const std::string &atomStyle)
```

Sets the default atom style. This function accepts either a string or an enumerated representation as its parameter. The default is "small". See the AtomStyleSetScoped function for more information about atom styles.

17.4.32 AtomStyleSetScoped

```
void AtomStyleSetScoped(unsigned int style,
                        unsigned int scope = BestScope)
void AtomStyleSetScoped(const std::string &atomStyle,
                        unsigned int scope = BestScope)
```

Sets the display style for the specified atoms. This function accepts either a string or an enumerated representation as its parameter. Available atom styles include:

```
* "small"   atoms are drawn as small sized spheres
* "medium"  atoms are drawn as medium sized spheres
* "large"   atoms are drawn as large sized spheres
* "stars"   atoms are drawn as stars
```

(continues on next page)

(continued from previous page)

```
* "radii"   atoms are drawn as spheres with radii equivalent to their VDW radii
* "none"    atoms are not drawn, but are still considered visible and thus selectable
* "hidden"  atoms are not drawn and are not considered visible and thus are not␣
↪selectable
```

17.4.33 AtomStyleToEnum

```
unsigned int AtomStyleToEnum(const std::string &style)
```

This function returns the enumerated type associated the specified string atoms style representation. See the AtomStyleSetScoped function for more information about atom styles.

17.4.34 AtomStyleToText

```
std::string AtomStyleToText(unsigned int style)
```

This function returns the text string associated with the specified enumerated atom style representation. See the AtomStyleSetScoped function for more information about atom styles.

17.4.35 BondBallToStickRatioGet

```
float BondBallToStickRatioGet()
```

This function is currently a placeholder for future functionality.

17.4.36 BondBallToStickRatioSet

```
float BondBallToStickRatioSet(float ballStickRatio)
```

This function is currently a placeholder for future functionality.

17.4.37 BondClearColorScoped

```
void BondClearColorScoped(unsigned int scope = BestScope)
```

This function restores the default bond coloring to those bonds within the specified scope.

17.4.38 BondColorSetScoped

```
void BondColorSetScoped(const OESystem::OEColor &color,
                        unsigned int scope = BestScope)
```

This function sets the color of the bonds in the specified scope to be the specified color *c*.

17.4.39 BondDrawAromaticGet

```
bool BondDrawAromaticGet(unsigned int id)
```

Returns whether or not aromatic bonds should be drawn as aromatic or instead in Kekule format for the molecule with the specified *id*.

17.4.40 BondDrawAromaticSet

```
bool BondDrawAromaticSet(unsigned int id, bool aromatic)
```

Sets whether or not aromatic bonds should be drawn as aromatic or instead in kekule format for the molecule with the specified *id*.

17.4.41 BondHideHydrogenGet

```
bool BondHideHydrogenGet()
```

Returns the global property which specifies whether or not explicit hydrogens should be displayed.

17.4.42 BondHideHydrogenSet

```
bool BondHideHydrogenSet(bool hide)
```

Sets the global property which specifies whether or not explicit hydrogens should be displayed.

17.4.43 BondHideNonbondedGet

```
bool BondHideNonbondedGet()
```

Returns the global property which specifies whether or not explicit non-bonded atoms should be displayed.

17.4.44 BondHideNonbondedSet

```
bool BondHideNonbondedSet(bool hide)
```

Sets the global property which specifies whether or not explicit non-bonded atoms should be displayed.

17.4.45 BondLabelDefaultGet

```
std::string BondLabelDefaultGet()
```

Returns the default label displayed on bonds. The default is “None”. See the BondLabelSetScoped function for more information about label types.

17.4.46 BondLabelDefaultSet

```
void BondLabelDefaultSet(const std::string &label)
```

Sets the default label displayed on bonds. The default is “None”. See the BondLabelSetScoped function for more information about label types.

17.4.47 BondLabelGet

```
std::string BondLabelGet(const OEPropDB::OEKey &k)
```

Returns the current label on the specified bond.

17.4.48 BondLabelSet

```
bool BondLabelSet(const OEPropDB::OEKey &k, const std::string &label)
```

Sets the label on the specified bond. The parameter *label* contains the text of the label and the parameter *key* specifies which bond this label should apply to. See the BondLabelSetScoped function for more information about label types.

17.4.49 BondLabelSetScoped

```
bool BondLabelSetScoped(const std::string &label,
                        unsigned int scope = BestScope)
```

Sets the current label displayed on the bonds in the specified scope. Label types include:

```
* "none"               or ""
* "index"              or "%i"
* "order"              or "%o"
* "length"             or "%l"
* "type"               or "%t"
* "inttype"            or "%it"
* "ischiral"           or "%c"
* "chirality"          or "%chi"
* "hasstereospecified" or "%sts"
* "oechem idx"         or "%oei"
* "%gd(x)" - where x specifies the desired piece of generic data to use as the label
```

17.4.50 BondLineWidthGet

```
float BondLineWidthGet()
```

Returns the line width (in pixels) to be used when drawing bonds in “line” or “wireframe” styles.

17.4.51 BondLineWidthSet

```
float BondLineWidthSet(float width)
bool BondLineWidthSet(const OEPropDB::OEKey &k, float width)
```

Sets the line width (in pixels) to be used when drawing bonds in “line” or “wireframe” styles.

17.4.52 BondRadiusGet

```
float BondRadiusGet()
```

Returns the radius of the cylinders to be used when drawing bonds in “cylinder” style.

17.4.53 BondRadiusSet

```
float BondRadiusSet(float radius)
bool BondRadiusSet(const OEPropDB::OEKey &k, float radius)
```

Sets the radius of the cylinders to be used when drawing bonds in “cylinder” style.

17.4.54 BondShowAromaticGet

```
bool BondShowAromaticGet()
```

Returns the global property which specifies whether or not aromatic bonds should be drawn as aromatic or instead in kekule format.

17.4.55 BondShowAromaticSet

```
bool BondShowAromaticSet(bool show)
```

Sets the global property which specifies whether or not aromatic bonds should be drawn as aromatic or instead in kekule format.

17.4.56 BondShowDipolarGet

```
bool BondShowDipolarGet()
```

Returns the global property which specifies whether or not dipolar bonds should be drawn as directed bonds.

17.4.57 BondShowDipolarSet

```
bool BondShowDipolarSet(bool show)
```

Sets the global property which specifies whether or not dipolar bonds should be drawn as directed bonds.

17.4.58 BondShowOrdersGet

```
bool BondShowOrdersGet()
```

Returns the global property which specifies whether or not bond orders should be displayed.

17.4.59 BondShowOrdersSet

```
bool BondShowOrdersSet(bool show)
```

Sets the global property which specifies whether or not bond orders should be displayed.

17.4.60 BondStyleGet

```
std::string BondStyleGet()
```

Returns the default bond style. The default is “cylinders”. See the BondStyleSetScoped function for more information about bond styles.

17.4.61 BondStyleGetScoped

```
unsigned int BondStyleGetScoped(unsigned int scope = BestScope)
```

Returns the style of the bonds in the specified scope. If there are multiple styles present, the return type is “mixed”. See the BondStyleSetScoped function for more information about bond styles.

17.4.62 BondStyleLargeGet

```
std::string BondStyleLargeGet()
```

Returns the default bond style for large molecules, where large is determined by whether or not the number of atoms in the molecule exceeds the specified cutoff. For more information on this cutoff, see the MoleculeSizeCutoff function. The default is "lines". See the BondStyleSetScoped function for more information about bond styles.

17.4.63 BondStyleLargeSet

```
void BondStyleLargeSet(unsigned int style)
void BondStyleLargeSet(const std::string &bondstyle)
```

Sets the default bond style for large molecules, where large is determined by whether or not the number of atoms in the molecule exceeds the specified cutoff. This function accepts either a string or an enumerated representation as its parameter. The default is "lines". For more information on this large molecule atom cutoff, see the MoleculeSizeCutoff function. See the BondStyleSetScoped function for more information about bond styles.

17.4.64 BondStyleNucleicGet

```
std::string BondStyleNucleicGet()
```

Returns the default bond style for nucleic acids. The default is to mirror the large molecule bond style. See the BondStyleLargeSet function for more information about large molecules and see the BondStyleSetScoped function for more information about bond styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.65 BondStyleNucleicSet

```
void BondStyleNucleicSet(unsigned int style)
void BondStyleNucleicSet(const std::string &bondstyle)
```

Sets the default bond style for nucleic acids. This function accepts either a string or an enumerated representation as its parameter. The default is to mirror the large molecule atom style. See the AtomStyleLargeSet function for more information about large molecules and see the AtomStyleSetScoped function for more information about atom styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.66 BondStyleProteinGet

```
std::string BondStyleProteinGet()
```

Returns the default bond style for proteins. The default is to mirror the large molecule bond style. See the BondStyleLargeSet function for more information about large molecules and see the BondStyleSetScoped function for more information about bond styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.67 BondStyleProteinSet

```
void BondStyleProteinSet(unsigned int style)
void BondStyleProteinSet(const std::string &bondstyle)
```

Sets the default bond style for proteins. The default is to mirror the large molecule bond style. See the BondStyleLargeSet function for more information about large molecules and see the BondStyleSetScoped function for more information about bond styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.68 BondStyleSet

```
void BondStyleSet(unsigned int style)
void BondStyleSet(const std::string &bondstyle)
```

Sets the default bond style. This function accepts either a string or an enumerated representation as its parameter. The default is "cylinders". See the BondStyleSetScoped function for more information about bond styles.

17.4.69 BondStyleSetScoped

```
void BondStyleSetScoped(unsigned int style,
                        unsigned int scope = BestScope)
void BondStyleSetScoped(const std::string &bondstyle,
                        unsigned int scope = BestScope)
```

Sets the display style for the specified bonds. This function accepts either a string or an enumerated representation as its parameter. Available bonds styles include:

```
* "lines"     bonds are drawn as lines
* "cylinders" bonds are drawn as cylinders
* "none"      bonds are not drawn, but are still considered visible and thus selectable
* "hidden"    bonds are not drawn and are not considered visible and thus are not␣
↪selectable
```

17.4.70 BondStyleToEnum

```
unsigned int BondStyleToEnum(const std::string &style)
```

This function returns the enumerated type associated the specified string bond style representation. See the BondStyleSetScoped function for more information about bond styles.

17.4.71 BondStyleToText

```
std::string BondStyleToText(unsigned int style)
```

This function returns the text string associated with the specified enumerated bond style representation. See the BondStyleSetScoped function for more information about bond styles.

17.4.72 BookmarkDelete

```
bool BookmarkDelete(const std::string &name)
```

Deletes the bookmark specified by *name*.

17.4.73 BookmarkExists

```
bool BookmarkExists(const std::string &name)
```

Returns whether or not the specified bookmark exists.

17.4.74 BookmarkLoad

```
bool BookmarkLoad(const std::string &name)
```

Loads the specified bookmark.

17.4.75 BookmarkMoveDown

```
bool BookmarkMoveDown(const std::string &name)
```

Moves the specified bookmark down in the order of the current set of bookmarks.

17.4.76 BookmarkMoveUp

```
bool BookmarkMoveUp(const std::string &name)
```

Moves the specified bookmark up in the order of the current set of bookmarks.

17.4.77 BookmarkOrganizeDialog

```
std::string BookmarkOrganizeDialog(bool exec=false)
```

Launches an interactive dialog which allows the user to delete, rename, and reorder the current set of bookmarks.

17.4.78 PromptBookmarkAnimationTime

```
int PromptBookmarkAnimationTime( int originalTime )
```

Launches an interactive dialog which allows the user to specify the animation duration when transitioning to a selected bookmark. Two options are provided:

```
* "Use calculated animation time"          automatically calculated using limits on␣
↪speed of rotation, scaling and translation.
* "Specify animation time in milliseconds" specified by the user.
```

17.4.79 BookmarkRename

```
bool BookmarkRename(const std::string &oldName, const std::string &newName)
```

Changes the name of the bookmark specified by *oldName* to the name specified by *newName*.

17.4.80 BookmarkSave

```
bool BookmarkSave(const std::string &name)
```

Saves the current state of the application to a named bookmark specified by the parameter *name*.

17.4.81 Bookmarks

```
std::vector<std::string> Bookmarks()
```

Returns a list of the current set of bookmarks.

17.4.82 BookmarksClear

```
void BookmarksClear()
```

Clears the current list of bookmarks.

17.4.83 BookmarksLastLoaded

```
unsigned int BookmarksLastLoaded()
```

Returns the index (in the order of the current set of bookmarks) of the last bookmark that was loaded.

17.4.84 CATraceRadiusGet

```
float CATraceRadiusGet()
```

Returns the global property which specifies the width of CA Traces.

17.4.85 CATraceRadiusSet

```
float CATraceRadiusSet(float radius)
```

Sets the global property which specifies the width of CA Traces.

17.4.86 CATraceStyleGet

```
std::string CATraceStyleGet()
std::string CATraceStyleGet(const OEPropDB::OEKey &k)
```

Returns the global property which specifies the style in which CA Traces are drawn. Available styles include: “lines” and “cylinders”.

17.4.87 CATraceStyleSet

```
std::string CATraceStyleSet(const std::string &style)
std::string CATraceStyleSet(const OEPropDB::OEKey &k, const std::string &style)
```

Sets the global property which specifies the style in which CA Traces are drawn. Available styles include: “lines” and “cylinders”.

17.4.88 CATraceStyleSetScoped

```
std::string CATraceStyleSetScoped(const std::string &str,
                                  unsigned int scope = BestScope)
```

Sets the property which specifies the style in which CA Traces are drawn. Available styles include: “lines” and “cylinders”.

17.4.89 CenterSet

```
void CenterSet(const OEPropDB::OEKey &)
void CenterSet(const std::vector<OEPropDB::OEKey> &)
```

Sets the center of the scene to be at the center of the object(s) referenced by the specified key(s).

17.4.90 CenterSetScoped

```
void CenterSetScoped(unsigned int scope)
```

Scoped version of *CenterSet*.

17.4.91 ColorGet

```
OESystem::OEColor ColorGet(const OEPropDB::OEKey &)
```

Returns the color of the object specified by the OEKey parameter *key*.

17.4.92 ColorSet

```
void ColorSet(const OEPropDB::OEKey &, const OESystem::OEColor &)
```

Sets the color of the object specified by the OEKey parameter *key* to be the specified *color*.

17.4.93 ColorSetScoped

```
void ColorSetScoped(const OESystem::OEColor &,
                    unsigned int scope = BestScope)
```

Sets the color for all of the objects in the specified scope to be the specified *color*.

17.4.94 ColorUniqueScoped

```
void ColorUniqueScoped(unsigned int scope = BestScope)
```

Assigns a "unique" color to each individual object found in the specified *scope*. For large numbers of objects in the scope, the colors used may eventually repeat.

17.4.95 ContourAutoCenterGet

```
bool ContourAutoCenterGet(unsigned int id=0)
```

Returns whether or not reentrant grids are automatically contoured at the center of the scene.

17.4.96 ContourAutoCenterSet

```
bool ContourAutoCenterSet(unsigned int id, bool center)
```

Sets whether or not reentrant grids are automatically contoured at the center of the scene. If this is set to true, translating the center of the scene will cause the grids to be recontoured as the scene moves.

17.4.97 ContourAutoContourGet

```
bool ContourAutoContourGet()
```

Returns whether or not reentrant grids are automatically recontoured when the center of the scene is changed.

17.4.98 ContourAutoContourRadiusScaleSet

```
bool ContourAutoContourRadiusScaleSet(float radius, float scale)
```

Sets the radius and scale parameters used in determining how to contour a reentrant grid around the center of the scene.

17.4.99 ContourAutoContourSet

```
void ContourAutoContourSet(bool autocontour)
```

Sets whether or not reentrant grids are automatically recontoured when the center of the scene is changed.

17.4.100 ContourCenter

```
std::vector<float> ContourCenter(unsigned int id=0)
std::vector<float> ContourCenter(unsigned int id, std::vector<float>)
```

Returns the scene center position associated with the specified contour.

17.4.101 ContourColorForIndexGet

```
OESystem::OEColor ContourColorForIndexGet(unsigned int id, unsigned int index)
```

Return the color of the grid contour specified by the grid *id* and contour *index*.

17.4.102 ContourColorForIndexGetScoped

```
OESystem::OEColor ContourColorForIndexGetScoped(unsigned int index,
                                                unsigned int scope = BestScope)
```

Returns the color of the grid contours in the specified scope at contour index *index*. This color is determined by looking at the first contour found in the specified scope.

17.4.103 ContourColorForIndexSet

```
OESystem::OEColor ContourColorForIndexSet(unsigned int id, unsigned int index,
                                          const OESystem::OEColor &color)
```

For grid *id* set the color of contour *index* to *color*.

17.4.104 ContourColorForIndexSetScoped

```
OESystem::OEColor ContourColorForIndexSetScoped(unsigned int index,
                                                const OESystem::OEColor &color,
                                                unsigned int scope = BestScope)
```

For all grids in *scope* set the color of contour *index* to *color*.

17.4.105 ContourColorSet

```
void ContourColorSet(const OEPropDB::OEKey &, const OESystem::OEColor &)
```

This function sets the color of the contour associated with the specified key.

17.4.106 ContourColorSetScoped

```
void ContourColorSetScoped(const OESystem::OEColor &,
                           unsigned int scope = BestScope)
```

This function sets the color of the contours in the specified scope.

17.4.107 ContourDrawAsSurfaceGet

```
bool ContourDrawAsSurfaceGet(unsigned int id=0)
```

Returns whether or not the contour should be drawn as a solid surface or as a mesh.

17.4.108 ContourDrawAsSurfaceSet

```
bool ContourDrawAsSurfaceSet(unsigned int id, bool draw)
```

Sets whether or not the contour should be drawn as a solid surface or as a mesh.

17.4.109 ContourDrawAsSurfaceSetScoped

```
void ContourDrawAsSurfaceSetScoped(bool draw,
                                   unsigned int scope = BestScope)
```

Returns the style of contour drawing for the surfaces in scope; essentially it returns the state of the first grid found in scope.

See *ContourDrawAsSurfaceGet*.

17.4.110 ContourDrawStyleGet

```
std::string ContourDrawStyleGet(unsigned int id=0)
```

If “surface” the contours are drawn as a surface, otherwise the contour is drawn as a mesh.

17.4.111 ContourDrawStyleSet

```
void ContourDrawStyleSet(unsigned int id, const std::string &)
```

If set to “surface” the contours are drawn as a surface, otherwise the contour is drawn as a mesh.

17.4.112 ContourDrawStyleSetScoped

```
void ContourDrawStyleSetScoped(const std::string &style,
                               unsigned int scope = BestScope)
```

Set the drawstyle to all contours in scope.

17.4.113 ContourEntireGridGet

```
bool ContourEntireGridGet(unsigned int id=0)
```

Returns True if grid *id* is contoured over the whole grid. Returns False if only the portion near the center of the scene is contoured.

17.4.114 ContourEntireGridSet

```
bool ContourEntireGridSet(unsigned int id, bool entire)
```

If true contour the entire extent of grid *id*, otherwise only contour the portion near the center of the drawing scene.

17.4.115 ContourHideIndex

```
void ContourHideIndex(unsigned int index, bool hide)
void ContourHideIndex(unsigned int id, unsigned int index, bool hide)
```

This function controls the visibility of the contour specified by the grid *id* and contour *index*. If the *hide* parameter is set to true, the contour is hidden, otherwise the contour is shown.

17.4.116 ContourHideIndexGet

```
bool ContourHideIndexGet(unsigned int id, unsigned int index)
```

This function returns whether or not the contour specified by the grid *id* and contour *index* is hidden or not.

17.4.117 ContourHideIndexSet

```
void ContourHideIndexSet(unsigned int index, bool hide)
void ContourHideIndexSet(unsigned int id, unsigned int index, bool hide)
```

This function controls the visibility of the contour specified by the grid *id* and contour *index*. If the *hide* parameter is set to true, the contour is hidden, otherwise the contour is shown.

17.4.118 ContourLineWidthGet

```
float ContourLineWidthGet(unsigned int id=0)
```

Returns the line width for drawing mesh contours.

17.4.119 ContourLineWidthSet

```
float ContourLineWidthSet(unsigned int id, float rad)
```

Set the line width to *width* for drawing mesh contours.

17.4.120 ContourPickIsoSurfacesGet

```
bool ContourPickIsoSurfacesGet()
```

Returns whether or not contours are selectable by picking.

17.4.121 ContourPickIsoSurfacesSet

```
void ContourPickIsoSurfacesSet(bool pickable)
```

Sets whether or not contours are selectable by picking.

17.4.122 ContourTransparencySet

```
void ContourTransparencySet(const OEPropDB::OEKey &, unsigned int)
```

Sets the transparency of the contour associated with the specified key. The transparency is determined by the *alpha* parameter which corresponds to the color alpha value which can range from 0 (completely transparent) to 255 (completely opaque).

17.4.123 CustomViewEON

```
void CustomViewEON()
void CustomViewEON(const std::string &results, const std::string &reference="")
void CustomViewEON(const std::string &results, const std::string &reference,
                   bool refIsFirst, bool tiled, bool showSpreadsheet,
                   bool showStyleControl,
                   unsigned int maxTiles)
```

This function sets up the display for optimal visualization of output from the application EON.

17.4.124 CustomViewFRED

```
void CustomViewFRED()
void CustomViewFRED(const std::string &protein, const std::string &reference,
                    const std::string &results="")
void CustomViewFRED(const std::string &protein, const std::string &reference,
                    const std::string &results, bool showRibbons, bool showLabels,
                    bool showHBonds, bool showSurface, float residueWithin,
                    float surfaceWithin, const std::string &surfaceColor)
```

This function sets up the display for optimal visualization of output from the application FRED.

17.4.125 CustomViewROCS

```
void CustomViewROCS()
void CustomViewROCS(const std::string &results,
                    const std::string &reference="")
void CustomViewROCS(const std::string &results, const std::string &reference,
                    bool refIsFirst, bool tiled, bool showSpreadsheet,
                    bool unused, bool showStyleControl,
                    unsigned int maxTiles)
```

This function sets up the display for optimal visualization of output from the application ROCS.

17.4.126 DefaultColorLabelGet

```
OESystem::OEColor DefaultColorLabelGet()
```

Returns the default color for a label.

17.4.127 DefaultColorLabelSet

```
OESystem::OEColor DefaultColorLabelSet(const OESystem::OEColor &)
```

Returns the default color for a label.

17.4.128 DefaultColorMarkedGet

```
OESystem::OEColor DefaultColorMarkedGet()
```

Returns the default color used when displaying marked objects.

17.4.129 DefaultColorMarkedSet

```
OESystem::OEColor DefaultColorMarkedSet(const OESystem::OEColor &)
```

Sets the default color used when displaying marked objects.

17.4.130 DefaultColorReferenceGet

```
OESystem::OEColor DefaultColorReferenceGet()
```

Returns the default reference molecule color.

17.4.131 DefaultColorReferenceSet

```
OESystem::OEColor DefaultColorReferenceSet(const OESystem::OEColor &)
```

Sets the default reference molecule color.

17.4.132 DefaultColorSelectedGet

```
OESystem::OEColor DefaultColorSelectedGet()
```

Returns the default selection color.

17.4.133 DefaultColorSelectedSet

```
OESystem::OEColor DefaultColorSelectedSet(const OESystem::OEColor &)
```

Sets the default selection color.

17.4.134 DefaultColorTitleGet

```
OESystem::OEColor DefaultColorTitleGet()
```

Returns the default title color.

17.4.135 DefaultColorTitleSet

```
OESystem::OEColor DefaultColorTitleSet(const OESystem::OEColor &)
```

Sets the default title color.

17.4.136 DefaultMonitorColorGet

```
OESystem::OEColor DefaultMonitorColorGet()
```

Returns the default monitor color.

17.4.137 DefaultMonitorColorSet

```
OESystem::OEColor DefaultMonitorColorSet(const OESystem::OEColor &)
```

Sets the default monitor color.

17.4.138 DistanceControlsVisibilityGet

```
bool DistanceControlsVisibilityGet()
```

Returns whether or not an atom's or bond's distance from the selected set will affect its visibility. The distance cutoff is specified by the AtomHaloSet function.

17.4.139 DistanceControlsVisibilitySet

```
void DistanceControlsVisibilitySet(bool state)
```

Sets whether or not an atom's or bond's distance from the selected set will affect its visibility. The distance cutoff is specified by the AtomHaloSet function.

17.4.140 DrawAtomsAndBondsGet

```
bool DrawAtomsAndBondsGet()
```

Returns whether or not the application has drawing of atoms and bonds enabled. Ordinarily, this function should not need to be called by the user.

17.4.141 DrawAtomsAndBondsSet

```
bool DrawAtomsAndBondsSet(bool draw)
```

Sets whether or not the application has drawing of atoms and bonds enabled. Ordinarily, this function should not need to be called by the user.

17.4.142 DrawAxesGet

```
bool DrawAxesGet()
```

Returns whether or not axes should be drawn in the application's 3D display.

17.4.143 DrawAxesSet

```
bool DrawAxesSet(bool draw)
```

Sets whether or not axes should be drawn in the application's 3D display.

17.4.144 DrawCATracesGet

```
bool DrawCATracesGet()
bool DrawCATracesGet(const OEPropDB::OEKey &k)
```

Returns whether or not the application has drawing of CA Traces enabled. Ordinarily, this function should not need to be called by the user.

17.4.145 DrawCATracesSet

```
bool DrawCATracesSet(bool draw)
bool DrawCATracesSet(const OEPropDB::OEKey &, bool draw)
```

Sets whether or not the application has drawing of CA Traces enabled. Ordinarily, this function should not need to be called by the user.

17.4.146 DrawCATracesSetScoped

```
bool DrawCATracesSetScoped(bool, unsigned int scope = BestScope)
```

Sets whether or not a CA Trace should be drawn the proteins in the specified scope.

17.4.147 DrawContoursGet

```
bool DrawContoursGet()
```

Returns true if the contours are being drawn. Returns false if no contours are being drawn.

17.4.148 DrawContoursSet

```
bool DrawContoursSet(bool draw)
```

If *draw* is true, then draw all contours for each visible grid. Otherwise no contours are drawn.

17.4.149 DrawLabelsGet

```
bool DrawLabelsGet()
```

Returns whether or not the application has drawing of labels enabled.

17.4.150 DrawLabelsSet

```
bool DrawLabelsSet(bool draw)
```

Sets whether or not the application has drawing of labels enabled.

17.4.151 DrawMatrixGet

```
bool DrawMatrixGet()
```

Returns whether or not the application has matrix (or tiled) displays enabled.

17.4.152 DrawMatrixSet

```
bool DrawMatrixSet(bool draw)
```

Sets whether or not the application has matrix (or tiled) displays enabled.

17.4.153 DrawRibbonsGet

```
bool DrawRibbonsGet()
bool DrawRibbonsGet(const OEPropDB::OEKey &k)
```

Returns whether or not the application has drawing of protein ribbons enabled. Ordinarily, this function should not need to be called by the user.

17.4.154 DrawRibbonsSet

```
bool DrawRibbonsSet(bool draw)
bool DrawRibbonsSet(const OEPropDB::OEKey &k, bool draw)
```

Sets whether or not the application has drawing of protein ribbons enabled. Ordinarily, this function should not need to be called by the user.

17.4.155 DrawRibbonsSetScoped

```
bool DrawRibbonsSetScoped(bool draw,
                          unsigned int scope = BestScope)
```

Sets whether or not a ribbon display should be shown for the proteins in the specified scope.

17.4.156 DrawSurfacesGet

```
bool DrawSurfacesGet()
```

Returns true if surfaces are being drawn, false if not.

17.4.157 DrawSurfacesSet

```
bool DrawSurfacesSet(bool draw)
```

If parameter *draw* if false, surface objects are not drawn. If *draw* is true surfaces are drawn.

17.4.158 DrawSymmetryGet

```
bool DrawSymmetryGet()
```

Returns whether or not the application has drawing of symmetry enabled.

17.4.159 DrawSymmetrySet

```
bool DrawSymmetrySet(bool draw)
```

Sets whether or not the application has drawing of symmetry enabled.

17.4.160 DrawTitlesGet

```
bool DrawTitlesGet()
```

Returns whether or not the application has drawing of titles enabled.

17.4.161 DrawTitlesSet

```
bool DrawTitlesSet(bool draw)
```

Sets whether or not the application has drawing of symmetry enabled.

17.4.162 DrawUnitCellGet

```
bool DrawUnitCellGet()
```

Returns whether or not the application has drawing of unit cells enabled.

17.4.163 DrawUnitCellSet

```
bool DrawUnitCellSet(bool draw)
```

Sets whether or not the application has drawing of symmetry enabled.

17.4.164 GridDefaultContourColorByIndexGet

```
OESystem::OEColor GridDefaultContourColorByIndexGet(unsigned int gridType,
                                                    unsigned int index)
```

Returns the default contour color for contour *index* for grids of type *gridType*.

17.4.165 GridDefaultContourColorByIndexSet

```
void GridDefaultContourColorByIndexSet(unsigned int gridType,
                                       unsigned int index,
                                       const OESystem::OEColor &color)
```

Set the default contour *color* for contour *index* for grids of type *gridType*.

17.4.166 GridDefaultDrawAsSurfaceGet

```
bool GridDefaultDrawAsSurfaceGet(unsigned int gridType)
```

Returns the default drawing style for contours of grid with type *gridType*.

If true, then the surface is drawn as a solid surface, otherwise the surface is drawn as a mesh.

17.4.167 GridDefaultDrawAsSurfaceSet

```
void GridDefaultDrawAsSurfaceSet(unsigned int gridType, bool asSurf)
```

Set the default drawing style *asSurf* for contours of grid with type *gridType*.

If *asSurf* is true, then the surface is drawn as a solid surface, otherwise the surface is drawn as a mesh.

17.4.168 GridShowCornersGet

```
bool GridShowCornersGet()
```

Returns true if grid corners are being shown false if not.

Grid corners are indicators of the extent of visible grids. They are useful for displaying uncontoured grids and showing whether grid space and molecules intersect.

17.4.169 GridShowCornersSet

```
void GridShowCornersSet(bool show)
```

Set *show* to true to show grid corners, false if not.

Grid corners are indicators of the extent of visible grids. They are useful for displaying uncontoured grids and showing whether grid space and molecules intersect.

17.4.170 GridShowLastMaskedGrid

```
unsigned int GridShowLastMaskedGrid(bool show=true)
```

Reserved for internal use.

17.4.171 HBondAddTarget

```
void HBondAddTarget(const OEPropDB::OEKey &key)
```

This function adds the molecule corresponding to the specified *key* to the list of targets to be considered when displaying hydrogen bonds. Calling this function on a molecule will cause hydrogens to be added to that molecule if they are not already present.

17.4.172 HBondAddTargetsScoped

```
void HBondAddTargetsScoped(unsigned int scope = BestScope)
```

This function adds of the molecules in the specified *scope* to the list of targets to be considered when displaying hydrogen bonds. Calling this function on a molecule will cause hydrogens to be added to that molecule if they are not already present.

17.4.173 HBondClearTargets

```
void HBondClearTargets()
```

This function clears the list of molecule targets to be considered when displaying hydrogen bonds.

17.4.174 HBondColorGet

```
OESystem::OEColor HBondColorGet()
```

Returns the current color being used when drawing hydrogen bonds.

17.4.175 HBondColorSet

```
void HBondColorSet(const OESystem::OEColor &)
```

Sets the color to be used when drawing hydrogen bonds.

17.4.176 HBondRemoveTarget

```
void HBondRemoveTarget(const OEPropDB::OEKey &key)
```

This function removes the molecule corresponding to the specified *key* from the list of molecule targets to be considered when displaying hydrogen bonds.

17.4.177 HBondRemoveTargetsScoped

```
void HBondRemoveTargetsScoped(unsigned int scope = BestScope)
```

This function removes all of the molecules in the specified *scope* from the list of molecule targets to be considered when displaying hydrogen bonds.

17.4.178 HBondShowExternalGet

```
bool HBondShowExternalGet()
```

Returns whether or not external hydrogen bonds (those between two separate molecules) should be displayed if present.

17.4.179 HBondShowExternalSet

```
void HBondShowExternalSet(bool show)
```

Sets whether or not external hydrogen bonds (those between two separate molecules) should be displayed if present.

17.4.180 HBondShowInternalGet

```
bool HBondShowInternalGet()
```

Returns whether or not internal hydrogen bonds should be displayed if present.

17.4.181 HBondShowInternalSet

```
void HBondShowInternalSet(bool show)
```

Sets whether or not internal hydrogen bonds should be displayed if present.

17.4.182 HaloColorDefaultGet

```
OESystem::OEColor HaloColorDefaultGet()
```

Returns the default color to be used when displaying halos.

17.4.183 HaloColorDefaultSet

```
void HaloColorDefaultSet(const OESystem::OEColor &c)
```

Sets the default color to be used when displaying halos.

17.4.184 HaloColorGet

```
OESystem::OEColor HaloColorGet(const OEPropDB::OEKey &)
```

Returns the color of the halo associated with the specified atom key.

17.4.185 HaloColorSet

```
void HaloColorSet(const OEPropDB::OEKey &k, const std::string &)
void HaloColorSet(const OEPropDB::OEKey &k, const OESystem::OEColor &c)
```

Sets the color of the halo associated with the specified atom key.

17.4.186 HaloColorSetScoped

```
void HaloColorSetScoped(const std::string &,
                        unsigned int scope = BestScope)
void HaloColorSetScoped(const OESystem::OEColor &,
                        unsigned int scope = BestScope)
```

Sets the color of the halos associated with the atoms in the specified scope.

17.4.187 HaloRadiusGet

```
float HaloRadiusGet(const OEPropDB::OEKey &)
```

Returns the radius of the halo associated with the specified atom key.

17.4.188 HaloRadiusSet

```
void HaloRadiusSet(const OEPropDB::OEKey &, float)
void HaloRadiusSet(const OEPropDB::OEKey &, const std::string &)
```

Sets the radius of the halo associated with the specified atom key.

17.4.189 HaloRadiusSetScoped

```
void HaloRadiusSetScoped(float, unsigned int scope = BestScope)
void HaloRadiusSetScoped(const std::string &,
                         unsigned int scope = BestScope)
```

Sets the radius of the halos associated with the atoms in the specified scope.

17.4.190 HaloScaleGet

```
float HaloScaleGet(const OEPropDB::OEKey &)
```

Returns the scaling factor for the size of the halo associated with the specified atom key.

17.4.191 HaloScaleSet

```
void HaloScaleSet(const OEPropDB::OEKey &, float)
void HaloScaleSet(const OEPropDB::OEKey &, const std::string &)
```

Sets the scaling factor for the size of the halo associated with the specified atom key.

17.4.192 HaloScaleSetScoped

```
void HaloScaleSetScoped(float, unsigned int scope = BestScope)
void HaloScaleSetScoped(const std::string &,
                        unsigned int scope = BestScope)
```

Sets the scaling factor for the size of the halo associated with the atoms in the specified scope.

17.4.193 HideNoneScoped

```
void HideNoneScoped(unsigned int scope = BestScope)
```

Makes visible any hidden atoms or bonds on all the molecules in the specified scope.

17.4.194 HideOthers

```
void HideOthers(const std::vector<OEPropDB::OEKey> &keys, bool value)
```

Hides everything except the objects specified by list of keys passed as a parameter to this function.

17.4.195 HideScoped

```
void HideScoped(bool value, unsigned int scope = BestScope)
```

Hides everything in the specified scope.

17.4.196 LabelClearColorScoped

```
void LabelClearColorScoped(unsigned int scope = BestScope)
```

This function restores the default label coloring to those labels within the specified scope.

17.4.197 LabelClearScoped

```
void LabelClearScoped(unsigned int scope = BestScope)
```

Clears the labels on all the atoms and bonds in the specified scope.

17.4.198 LabelColorScoped

```
void LabelColorScoped(const OESystem::OEColor &c,
                      unsigned int scope = BestScope)
```

This function sets the color of the labels in the specified scope to be the specified color *c*.

17.4.199 LabelDefaultColorGet

```
OESystem::OEColor LabelDefaultColorGet()
```

This function returns the default color for labels.

17.4.200 LabelDefaultColorSet

```
void LabelDefaultColorSet(const OESystem::OEColor &c)
```

This function sets the default color for labels.

17.4.201 LabelFixedSizeGet

```
bool LabelFixedSizeGet()
```

Returns whether or not labels are drawn using a fixed size on the screen or whether they scale with zooming of the scene.

17.4.202 LabelFixedSizeSet

```
void LabelFixedSizeSet(bool fixed)
```

Sets whether or not labels are drawn using a fixed size on the screen or whether they scale with zooming of the scene.

17.4.203 LabelGet

```
std::string LabelGet(const OEPropDB::OEKey &key, bool full=false,
                     bool coords=true)
```

Returns a description of the specified object. If the *full* parameter is True, it will also include information from the parent object (if one exists). The *coords* parameter specifies whether or not coordinates should be included in the description (primarily for atoms and vertices).

17.4.204 MoleculeAltLocationShow

```
bool MoleculeAltLocationShow(const OEPropDB::OEKey &, unsigned int, bool)
bool MoleculeAltLocationShow(const OEPropDB::OEKey &, unsigned int, unsigned int,
                             unsigned int)
```

Sets the visibility of the specified alternate locations for the specified molecule. This function is primarily intended for internal use only.

17.4.205 MoleculeAltLocationVisible

```
bool MoleculeAltLocationVisible(const OEPropDB::OEKey &, unsigned int)
bool MoleculeAltLocationVisible(const OEPropDB::OEKey &, unsigned int,
                                unsigned int)
```

Returns whether or the specified alternate locations on the specified molecule are visible. This function is primarily intended for internal use only.

17.4.206 MoleculeAtomBondStyleSetScoped

```
void MoleculeAtomBondStyleSetScoped(const std::string &atomStyle,
                                    const std::string &bondStyle,
                                    unsigned int scope = BestScope)
```

Set the combined style for atoms and bonds.

- atomStyle is the style to set the atom , see AtomStyleSetScoped.
- bondStyle is the style to set the bond, see BondStyleSetScoped.
- scope is the default scope to use.

17.4.207 MoleculeColorByScoped

```
void MoleculeColorByScoped(const std::string &property,
                           const std::string &params="",
                           unsigned int scope = BestScope)
```

This color colors all the molecules in the specified scope according to the scheme specified by the *property* parameter. Valid properties include:

```
* "amino"
* "bfactor"
* "chain"
* "charge"
* "cpk"
* "cpknew"
* "formal charge"
* "group"
* "hbond"
* "hydrophobicity"
* "partial charge"
* "shapely"
```

If the specified property is *hydrophobicity*, the specific scale to be used can be specified in the *params* parameter. Valid scale options include:

```
* "charifson"
* "eisenberg"
* "kytedolittle"
* "whiteoctanol"
```

For more details on the specifics of these coloring scheme, please see the user manual.

17.4.208 MoleculeColorsResetScoped

```
void MoleculeColorsResetScoped(unsigned int scope = BestScope)
```

This function restores the default atom and bond coloring to those atoms and bonds in the specified scope. It is essentially equivalent to calling AtomClearColorScoped followed by BondClearColorScoped.

17.4.209 MoleculeDarkColorsGet

```
bool MoleculeDarkColorsGet()
```

Returns whether or not the application is using either the atom or residue color palette designed for dark colored backgrounds. Otherwise, the application is using the atom color palettes designed for light colored backgrounds. All of these palettes can be edited in the preferences. For more information, see the functions AtomDarkColorsGet and ResidueDarkColorsGet.

17.4.210 MoleculeDarkColorsSet

```
void MoleculeDarkColorsSet(bool dark)
```

Sets whether or not the application is using the atom and residue color palettes designed for dark colored backgrounds. Otherwise, the application is using the atom color palettes designed for light colored backgrounds. All of these palettes can be edited in the preferences. For more information, see the functions AtomDarkColorsSet and ResidueDarkColorsSet.

17.4.211 MoleculeShowfAntsyGet

```
bool MoleculeShowfAntsyGet()
```

Returns whether or not the application is using the high quality molecule representations as opposed to standard models.

17.4.212 MoleculeShowfAntsySet

```
bool MoleculeShowfAntsySet(bool show)
```

Sets whether or not the application is using the high quality molecule representations as opposed to standard models.

17.4.213 MoleculeStyleGet

```
std::string MoleculeStyleGet()
```

Returns the default molecule style. The default is “stick”. See the MoleculeStyleSetScoped function for more information about molecule styles.

17.4.214 MoleculeStyleGetScoped

```
unsigned int
  MoleculeStyleGetScoped(unsigned int scope = BestScope)
```

Returns the style of the molecules in the specified scope. If there are multiple styles present, the return type is “mixed”. See the MoleculeStyleSetScoped function for more information about molecule styles.

17.4.215 MoleculeStyleLargeGet

```
std::string MoleculeStyleLargeGet()
```

Returns the default molecule style for large molecules, where large is determined by whether or not the number of atoms in the molecule exceeds the specified cutoff. For more information on this cutoff, see the MoleculeSizeCutoff function. The default is "wireframe". See the MoleculeStyleSetScoped function for more information about molecule styles.

17.4.216 MoleculeStyleLargeSet

```
void MoleculeStyleLargeSet(unsigned int style)
void MoleculeStyleLargeSet(const std::string &style)
```

Sets the default molecule style for large molecules, where large is determined by whether or not the number of atoms in the molecule exceeds the specified cutoff. For more information on this cutoff, see the MoleculeSizeCutoff function. The default is "wireframe". See the MoleculeStyleSetScoped function for more information about molecule styles.

17.4.217 MoleculeStyleNucleicGet

```
std::string MoleculeStyleNucleicGet()
```

Returns the default molecule style for nucleic acids. The default is to mirror the large molecule style. See the MoleculeStyleLargeSet function for more information about large molecules and see the MoleculeStyleSetScoped function for more information about molecule styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.218 MoleculeStyleNucleicSet

```
void MoleculeStyleNucleicSet(unsigned int style)
void MoleculeStyleNucleicSet(const std::string &style)
```

Sets the default molecule style for nucleic acids. The default is to mirror the large molecule style. See the MoleculeStyleLargeSet function for more information about large molecules and see the MoleculeStyleSetScoped function for more information about molecule styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.219 MoleculeStyleProteinGet

```
std::string MoleculeStyleProteinGet()
```

Returns the default molecule style for proteins. The default is to mirror the large molecule style. See the MoleculeStyleLargeSet function for more information about large molecules and see the MoleculeStyleSetScoped function for more information about molecule styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.220 MoleculeStyleProteinSet

```
void MoleculeStyleProteinSet(unsigned int style)
void MoleculeStyleProteinSet(const std::string &style)
```

Sets the default molecule style for proteins. The default is to mirror the large molecule style. See the MoleculeStyleLargeSet function for more information about large molecules and see the MoleculeStyleSetScoped function for more information about molecule styles.

Note: this function is a placeholder for a future implementation and therefore does not currently have any effect.

17.4.221 MoleculeStyleSet

```
void MoleculeStyleSet(unsigned int style)
void MoleculeStyleSet(const std::string &atomStyle)
```

Sets the default molecule style. This function accepts either a string or an enumerated representation as its parameter. The default is "stick". See the MoleculeStyleSetScoped function for more information about molecule styles.

17.4.222 MoleculeStyleSetScoped

```
void MoleculeStyleSetScoped(unsigned int style,
                            unsigned int scope = BestScope)
void MoleculeStyleSetScoped(const std::string &atomStyle,
                            unsigned int scope = BestScope)
```

Sets the display style for the specified molecules. This function accepts either a string or an enumerated representation as its parameter. Available molecule styles include:

```
* "ball and stick" molecules are drawn in ball and stick mode
* "cpk"            molecules are drawn in CPK mode
* "stars"          molecules are drawn in star mode
* "stick"          molecules are drawn in stick mode
* "wireframe"      molecules are drawn in wireframe mode
* "none"           molecules are not drawn, but are still considered visible and thus␣
↪selectable
* "hidden"         molecules are not drawn and are not considered visible and thus are␣
↪not selectable
```

17.4.223 MoleculeStyleToEnum

```
unsigned int MoleculeStyleToEnum(const std::string &style)
```

This function returns the enumerated type associated the specified string molecule style representation. See the MoleculeStyleSetScoped function for more information about molecule styles.

17.4.224 MoleculeStyleToText

```
std::string MoleculeStyleToText(unsigned int style)
```

This function returns the text string associated with the specified enumerated molecule style representation. See the MolStyleSetScoped function for more information about molecule styles.

17.4.225 MonitorAngleCreate

```
unsigned int MonitorAngleCreate(const OEPropDB::OEKey &key1,
                                const OEPropDB::OEKey &key2,
                                const OEPropDB::OEKey &key3)
```

Attempts to create an angle monitor between the atoms specified by the parameters *key1*, *key2*, and *key3*.

Returns the ID of the created monitor or 0 if one could not be created.

17.4.226 MonitorAngleDelete

```
void MonitorAngleDelete(const OEPropDB::OEKey &key1,
                        const OEPropDB::OEKey &key2,
                        const OEPropDB::OEKey &key3)
```

Deletes the monitor, if any, specified by atom keys: *key1*, *key2*, *key3*.

17.4.227 MonitorAngleExists

```
unsigned int MonitorAngleExists(const OEPropDB::OEKey &key1,
                                const OEPropDB::OEKey &key2,
                                const OEPropDB::OEKey &key3)
```

Returns the ID of the angle monitor defined by the specified atom keys or zero if no such monitor exists.

17.4.228 MonitorColorGet

```
OESystem::OEColor MonitorColorGet(const OEPropDB::OEKey &monitor)
```

Returns the color of the specified monitor.

17.4.229 MonitorColorSet

```
void MonitorColorSet(const OEPropDB::OEKey &monitor,
                     const OESystem::OEColor &clr)
```

Sets the color of the specified monitor.

17.4.230 MonitorDeleteScoped

```
void MonitorDeleteScoped(unsigned int scope = BestScope)
```

Deletes all the monitors in the specified scope.

17.4.231 MonitorDistanceCreate

```
unsigned int MonitorDistanceCreate(const OEPropDB::OEKey &key1,
                                   const OEPropDB::OEKey &key2)
```

Attempts to create a distance monitor between the two atoms specified by the parameters *key1* and *key2*.

Returns the ID of the created monitor or 0 if one could not be created.

17.4.232 MonitorDistanceDelete

```
void MonitorDistanceDelete(const OEPropDB::OEKey &key1,
                           const OEPropDB::OEKey &key2)
```

Deletes the monitor specified by the two atom keys.

17.4.233 MonitorDistanceExists

```
unsigned int MonitorDistanceExists(const OEPropDB::OEKey &key1,
                                   const OEPropDB::OEKey &key2)
```

Returns the ID of the distance monitor determined by the two atom keys or zero if no such monitor exists.

17.4.234 MonitorSphereCreate

```
unsigned int MonitorSphereCreate(const OEPropDB::OEKey &key1,
                                 const std::string &name,
                                 const OESystem::OEColor &clr, float radius=0)
unsigned int MonitorSphereCreate(const OEPropDB::OEKey &parent,
                                 const std::string &name, float x, float y,
                                 float z, float radius,
                                 const OESystem::OEColor &color)
```

Create a sphere monitor on an atom specified by *key1* or at the coordinates specified by *x*, *y*, and *z*. The monitor is only visible when the specified atom key or parent key (in the case of coordinate specification) is visible.

Returns the ID of the newly created monitor or 0 if one could not be created.

17.4.235 MonitorTorsionCreate

```
unsigned int MonitorTorsionCreate(const OEPropDB::OEKey &key1,
                                  const OEPropDB::OEKey &key2,
                                  const OEPropDB::OEKey &key3,
                                  const OEPropDB::OEKey &key4)
```

Attempts to create a torsion monitor between the atoms specified by *key1*, *key2*, *key3*, and *key4*.

Returns the ID of the created monitor or 0 if one could not be created.

17.4.236 MonitorTorsionDelete

```
void MonitorTorsionDelete(const OEPropDB::OEKey &key1,
                          const OEPropDB::OEKey &key2,
                          const OEPropDB::OEKey &key3,
                          const OEPropDB::OEKey &key4)
```

Deletes the torsion monitor, if any, specified by the atom keys: *key1*, *key2*, *key3*, *key4*.

17.4.237 MonitorTorsionExists

```
unsigned int MonitorTorsionExists(const OEPropDB::OEKey &key1,
                                  const OEPropDB::OEKey &key2,
                                  const OEPropDB::OEKey &key3,
                                  const OEPropDB::OEKey &key4)
```

Returns the ID of the torsion monitor determined by the specified four atom keys or zero if no such monitor exists.

17.4.238 MonitorsVisible

```
bool MonitorsVisible()
```

This function returns whether any monitors are currently visible.

17.4.239 MonitorsVisibleDelete

```
void MonitorsVisibleDelete()
```

Deletes all monitors that are currently being displayed.

17.4.240 PaneActivated

```
void PaneActivated(unsigned int pane)
```

In matrix (tiled) mode, sets the specified pane to be the active pane in the display. Ordinarily, this function should not need to be called by the user.

17.4.241 ProteinColorByBFactor

```
void ProteinColorByBFactor(const OEPropDB::OEKey &key)
```

Colors a protein referenced by key. Atoms with the lowest temperature factors will be colored bright blue, while those with the highest will be colored bright red.

17.4.242 ProteinColorByBFactorScoped

```
void ProteinColorByBFactorScoped(unsigned int scope, int bins=10)
void ProteinColorByBFactorScoped(unsigned int scope,
                                 const std::vector<BFactorColor> &clrs)
void ProteinColorByBFactorScoped(unsigned int scope, float min, float max,
                                 const OESystem::OEColor &clr)
```

Scoped versions of *ProteinColorByBFactor*.

17.4.243 ResidueColorPaletteUpdate

```
void ResidueColorPaletteUpdate()
```

This function updates the default residue coloring palette based on whether or not the application is using a dark or light colored background scheme. This function is called internally and should not by called by the user.

17.4.244 ResidueDarkColorsGet

```
bool ResidueDarkColorsGet()
```

Returns whether or not the application is using the residue color palette designed for dark colored backgrounds. Otherwise, the application is using the residue color palette designed for light colored backgrounds. Both of these palettes can be edited in the preferences.

17.4.245 ResidueDarkColorsSet

```
void ResidueDarkColorsSet(bool dark)
```

Sets whether or not the application is using the residue color palette designed for dark colored backgrounds. Otherwise, the application is using the residue color palette designed for light colored backgrounds. Both of these palettes can be edited in the preferences.

17.4.246 ResidueDefaultColorGet

```
OESystem::OEColor ResidueDefaultColorGet()
OESystem::OEColor ResidueDefaultColorGet(unsigned int res)
```

Returns the current default residue color. The default residue color is applied to every residue that does not have its own specific default color assigned. Calling this function with the *res* parameter returns the specific default color associated with the passed residue index.

17.4.247 ResidueDefaultColorSet

```
void ResidueDefaultColorSet(const OESystem::OEColor &c)
void ResidueDefaultColorSet(unsigned int res, const OESystem::OEColor &c)
```

Sets the current default residue color. The default residue color is applied to every residue that does not have its own specific default color assigned. Calling this function with the *res* parameter sets the specific default color associated with the passed residue index.

17.4.248 RibbonClearColorScoped

```
void RibbonClearColorScoped(unsigned int scope = BestScope)
```

This function restores the default ribbon coloring to those ribbons within the specified *scope*.

17.4.249 RibbonColorSetScoped

```
void RibbonColorSetScoped(const OESystem::OEColor &c,
                          unsigned int scope = BestScope)
```

This function sets the color of the ribbons in the specified *scope* to be the specified color *c*.

17.4.250 RibbonCrossResolutionGet

```
unsigned int RibbonCrossResolutionGet()
```

Returns the cross resolution used in calculating protein ribbons.

17.4.251 RibbonCrossResolutionSet

```
unsigned int RibbonCrossResolutionSet(unsigned int res)
```

Sets the cross resolution used in calculating protein ribbons.

17.4.252 RibbonGapGet

```
float RibbonGapGet()
```

Returns the gap value used in calculating protein ribbons.

17.4.253 RibbonGapSet

```
float RibbonGapSet(float gap)
```

Sets the gap value used in calculating protein ribbons.

17.4.254 RibbonHeightScaleGet

```
float RibbonHeightScaleGet()
```

Returns the height scale used in calculating protein ribbons.

17.4.255 RibbonHeightScaleSet

```
float RibbonHeightScaleSet(float scale)
```

Sets the height scale used in calculating protein ribbons.

17.4.256 RibbonRadiusGet

```
float RibbonRadiusGet()
```

Returns the radius used in calculating protein ribbons.

17.4.257 RibbonRadiusSet

```
float RibbonRadiusSet(float radius)
```

Sets the radius used in calculating protein ribbons.

17.4.258 RibbonResolutionGet

```
unsigned int RibbonResolutionGet()
```

Returns the resolution used in calculating protein ribbons.

17.4.259 RibbonResolutionSet

```
unsigned int RibbonResolutionSet(unsigned int res)
```

Sets the resolution used in calculating protein ribbons.

17.4.260 RibbonSplineTypeGet

```
std::string RibbonSplineTypeGet()
```

Returns the spline type used in calculating protein ribbons. Available types include: “CubicHermite” and “Beta”.

17.4.261 RibbonSplineTypeSet

```
std::string RibbonSplineTypeSet(const std::string &type)
```

Sets the spline type used in calculating protein ribbons. Available types include: “CubicHermite” and “Beta”.

17.4.262 RibbonStyleGet

```
std::string RibbonStyleGet()
```

Returns the style used in displaying protein ribbons. Available types include: “Round”, “Lines”, and “Cartoon”. The default is “Cartoon”.

17.4.263 RibbonStyleSet

```
std::string RibbonStyleSet(const std::string &style)
std::string RibbonStyleSet(const OEPropDB::OEKey &k, const std::string &style)
```

Sets the style used in displaying protein ribbons. Available types include: “Round”, “Lines”, and “Cartoon”. The default is “Cartoon”.

17.4.264 RibbonStyleSetScoped

```
std::string RibbonStyleSetScoped(const std::string &style,
                                 unsigned int scope = BestScope)
```

Sets the style used in displaying ribbons for the proteins in the specified scope. Available types include: “Round”, “Lines”, and “Cartoon”. The default is “Cartoon”.

17.4.265 RibbonWidthScaleGet

```
float RibbonWidthScaleGet()
```

Returns the width scale used in calculating protein ribbons.

17.4.266 RibbonWidthScaleSet

```
float RibbonWidthScaleSet(float scale)
```

Sets the width scale used in calculating protein ribbons.

17.4.267 SceneDrawActiveBorderGet

```
bool SceneDrawActiveBorderGet()
```

Returns whether or not a border is drawn (in matrix mode) around the cell containing the focused object.

17.4.268 SceneDrawActiveBorderSet

```
void SceneDrawActiveBorderSet(bool draw)
```

Sets whether or not a border is drawn (in matrix mode) around the cell containing the focused object.

17.4.269 SceneMatrixModeGet

```
std::string SceneMatrixModeGet()
```

Returns the display style for the individual cells when the application is in matrix mode. Available styles are: "OnePerView", "LockedOnAll", and "LockedPerView".

17.4.270 SceneMatrixModeSet

```
std::string SceneMatrixModeSet(std::string mode)
```

Sets the display style for the individual cells when the application is in matrix mode. Available styles are: "OnePerView", "LockedOnAll", and "LockedPerView".

17.4.271 SelectionColorBlendFactorGet

```
float SelectionColorBlendFactorGet()
```

Returns the alpha buffer coloring blending factor for selection colors. The default value is 1.0 which corresponds to zero blending with the selection color being the one displayed. A value of 0.0 also corresponds to zero blending but with the underlying color being the one displayed. A value of 0.5 corresponds to even blending of the underlying color and the selected color.

17.4.272 SelectionColorBlendFactorSet

```
float SelectionColorBlendFactorSet(float alpha)
```

Sets the alpha buffer coloring blending factor for selection colors. The default value is 1.0 which corresponds to zero blending with the selection color being the one displayed. A value of 0.0 also corresponds to zero blending but with the underlying color being the one displayed. A value of 0.5 corresponds to even blending of the underlying color and the selected color.

17.4.273 ShowESGridScoped

```
void ShowESGridScoped(bool show=true,
                      unsigned int scope = BestScope)
```

If *show* is true, attach and show electrostatic grids to all molecules in *scope*.

If *show* if false, hide all attached electrostatic grids to molecules in *scope*.

17.4.274 ShowSurface

```
void ShowSurface(const OEPropDB::OEKey &, const std::string &, bool show)
```

Sets whether or not a property surface of the specified *type* is displayed for the molecule associated with the specified key. Valid types are "molecular" and "accessible".

17.4.275 ShowSurfaceScoped

```
void ShowSurfaceScoped(const std::string &type, bool show,
                       unsigned int scope = BestScope)
```

Scoped version of *ShowSurface*.

17.4.276 SurfaceAlterTransparency

```
void SurfaceAlterTransparency(unsigned int id, unsigned int alpha)
```

Sets the transparency of the surface associated with the specified *id*. The transparency is specified by the *alpha* parameter which determines the alpha component of the surface color. The alpha values ranges from 0 (completely transparent) to 255 (completely opaque).

17.4.277 SurfaceColorBy

```
std::string SurfaceColorBy(const std::string &mode="")
void SurfaceColorBy(const OEPropDB::OEKey &key, const std::string &mode,
                    const std::string &params="")
```

This function colors the surface referenced by the specified *key* using the specified *style*. Valid styles include:

- **atom - colors the surface based on the color of the nearest atom to the** vertex being colored
- concavity - colors the surface by concavity, red is high concavity.
- curvature - colors the surface by curvature, green is positive curvature.
- **distance - colors the surface by its distance to the currently selected atoms.** Each color band represents two angstroms.
- **electrostatics - colors the surface using a red-to-blue scheme based on the** electrostatic potential observed at the surface.
- **hydrogen bonds - colors the surface by the underlying atom hydrogen-bond** donor/acceptor properties. Donors are red and acceptors are blue.
- **hydrophobicity - colors the surface using a hydrophobic color scale. There** are four different hydrophobic scales available: charifson, eisenberg, kytedolittle, and whiteoctanol. The desired scale can be set in the application preferences.
- **potential - colors the surface based using a red-to-blue scheme (low to high)** over the values in the surface's potential array. This is not the same as coloring by electrostatics.

17.4.278 SurfaceColorByScoped

```
void SurfaceColorByScoped(const std::string &colorby,
                          const std::string &params="",
                          unsigned int scope = BestScope)
```

Scoped version of *SurfaceColorBy*.

17.4.279 SurfaceColorGet

```
OESystem::OEColor SurfaceColorGet(unsigned int id)
```

Returns the color of the specified surface.

17.4.280 SurfaceColorGetScoped

```
OESystem::OEColor
  SurfaceColorGetScoped(unsigned int scope = BestScope)
```

Returns the color of the first surface found in the specified scope.

17.4.281 SurfaceColorSet

```
OESystem::OEColor SurfaceColorSet(unsigned int id,
                                  const OESystem::OEColor &color)
```

Set the color of the surface *id* to *color*.

17.4.282 SurfaceColorSetScoped

```
void SurfaceColorSetScoped(const OESystem::OEColor &color,
                           unsigned int scope = BestScope)
```

Set the color of the surfaces in *scope* to *color*.

17.4.283 SurfaceGrowTriangle

```
void SurfaceGrowTriangle(unsigned int oerid, const OEPropDB::OEKey &trikey,
                         bool addtriangle=false)
```

Grow the surface selection when a triangle is double clicked. Ordinarily this function does not need to be called.

17.4.284 SurfaceLineWidthGet

```
float SurfaceLineWidthGet(unsigned int id=0)
```

Returns the line width used when drawing the surface specified by *id* in mesh mode.

17.4.285 SurfaceLineWidthSet

```
float SurfaceLineWidthSet(unsigned int id, float width)
```

Sets the line width used when drawing the surface specified by *id* in mesh mode.

17.4.286 SurfaceStyleGet

```
std::string SurfaceStyleGet(unsigned int id)
```

Returns the default style of surfaces. The style is one of:

- mesh - the surfaces are drawn as meshes.
- points - the surfaces are drawn as points.
- solid - the surfaces are drawn as solid objects.

17.4.287 SurfaceStyleGetScoped

```
std::string SurfaceStyleGetScoped(unsigned int scope = BestScope)
```

Return the style for the surfaces in scope. The value returned is from the first surface found in scope and is not indicative of all surfaces.

See *SurfaceStyleGet* for a list of surface styles.

17.4.288 SurfaceStyleSet

```
std::string SurfaceStyleSet(unsigned int id, const std::string &style)
```

Set the surface *id* to on of the following styles:

- mesh - the surface is drawn as a mesh.
- points - the surface is drawn as points.
- solid - the surface is drawn as a solid object.

17.4.289 SurfaceStyleSetScoped

```
void SurfaceStyleSetScoped(const std::string &style,
                           unsigned int scope = BestScope)
```

Scoped version of *SurfaceStyleSet*.

17.4.290 SurfaceTransparencySet

```
void SurfaceTransparencySet(unsigned int id, unsigned int alpha)
```

Set the transparency to all surfaces in *scope* to transparency.

17.4.291 SurfaceTransparencySetScoped

```
void SurfaceTransparencySetScoped(unsigned int transparency,
                                  unsigned int scope = BestScope)
```

Set the transparency to all surfaces in *scope* to transparency.

17.4.292 SurfaceVertexFloodScoped

```
int SurfaceVertexFloodScoped(unsigned int scope = BestScope)
```

Get the surface vertex flood of all surfaces in scope. This is essentially the flood of the focused surface or the surface of the focused or first visible molecule.

17.4.293 SymmetryColorModeGet

```
unsigned int SymmetryColorModeGet()
```

17.4.294 SymmetryColorModeSet

```
unsigned int SymmetryColorModeSet(unsigned int mode)
```

17.4.295 TitleDefaultColorGet

```
OESystem::OEColor TitleDefaultColorGet()
```

Returns the default color for titles.

17.4.296 TitleDefaultColorSet

```
void TitleDefaultColorSet(const OESystem::OEColor &c)
```

Sets the default color for titles.

17.4.297 TitlesDrawAboveGet

```
bool TitlesDrawAboveGet()
```

Returns whether or not titles are drawn at the top of the scene or the bottom of the scene.

17.4.298 TitlesDrawAboveSet

```
void TitlesDrawAboveSet(bool top)
```

Sets whether or not titles are drawn at the top of the scene or the bottom of the scene.

17.4.299 TransparencySet

```
void TransparencySet(const OEPropDB::OEKey &, unsigned int alpha)
```

This function sets the transparency of the object associated with the specified key. The transparency is determined by the specified parameter *alpha* which corresponds to the desired alpha value of the object's color. Alpha can range from 0 (completely transparent) to 255 (completely opaque).

17.4.300 TransparencySetScoped

```
void TransparencySetScoped(unsigned int alpha,
                           unsigned int scope = BestScope)
```

This function sets the transparency of all the objects in the specified scope. The transparency is determined by the specified parameter *alpha* which corresponds to the desired alpha value of the object's color. Alpha can range from 0 (completely transparent) to 255 (completely opaque).

17.4.301 ViewerAmbientLightGet

```
OESystem::OEColor ViewerAmbientLightGet()
```

Returns the OpenGL ambient light property.

17.4.302 ViewerAmbientLightSet

```
OESystem::OEColor ViewerAmbientLightSet(const OESystem::OEColor &color)
```

Sets the OpenGL ambient light property.

17.4.303 ViewerAmbientMaterialGet

```
OESystem::OEColor ViewerAmbientMaterialGet()
```

Returns the OpenGL ambient material property.

17.4.304 ViewerAmbientMaterialSet

```
OESystem::OEColor ViewerAmbientMaterialSet(const OESystem::OEColor &color)
```

Sets the OpenGL ambient material property.

17.4.305 ViewerAnimate

```
void ViewerAnimate(const std::vector<float> &centers,
                   const std::vector<float> &look,
                   const std::vector<float> &up,
                   unsigned int              msec,
                   unsigned int              fps)
```

Reserved for future use.

17.4.306 ViewerAnimateTo

```
void ViewerAnimateTo( const std::string &bookmark,
                      unsigned int       msec,
                      unsigned int       fps)
void ViewerAnimateTo( const OEPropDB::OEKey &centerKey,
                      const OEPropDB::OEKey &refKey,
                      unsigned int           msec,
                      unsigned int           fps,
                      float                  zoom = 0.0 )
void ViewerAnimateTo( const std::vector<float> &center,
                      const std::vector<float> &look,
                      const std::vector<float> &up,
                      unsigned int              msec,
                      unsigned int              fps,
                      float                     zoom = 0.0 )
```

Animates the scene from the current position to the specified position. All implementations of this function take an *msec* and *fps* parameters. The *msec* parameter specifies how long it should take the animation to be performed. Please note that this time is only a suggestion, as many factors including CPU speed, GPU speed, and available memory will influence the actual performance of the animation. The *fps* parameters specifies how many frames per second should be used in the animation.

The first implementation takes a bookmark name as its only other parameter. The second implementation takes two OEKey parameters corresponding to the desired final center and viewing point positions for the scene. The third

implementation takes three arrays corresponding to the center coordinates, a vector defining the viewing angle, and a vector defining the up direction for the scene.

17.4.307 ViewerAntialiasGet

```
bool ViewerAntialiasGet()
```

Returns whether or not antialiasing is turned on for line drawing.

17.4.308 ViewerAntialiasSet

```
bool ViewerAntialiasSet(bool state)
```

Sets whether or not antialiasing is turned on for line drawing.

17.4.309 ViewerAutoCenterGet

```
bool ViewerAutoCenterGet()
```

Returns whether or not the scene will automatically be recentered when a change in the scene occurs (e.g. a new object is made active, objects are hidden or shown).

17.4.310 ViewerAutoCenterPanesGet

```
bool ViewerAutoCenterPanesGet()
```

Returns whether or not the scene in each pane in multi-pane display mode is automatically centered or whether all panes observe the same center.

17.4.311 ViewerAutoCenterPanesSet

```
void ViewerAutoCenterPanesSet(bool center)
```

Sets whether or not the scene in each pane in multi-pane display mode is automatically centered or whether all panes observe the same center.

17.4.312 ViewerAutoCenterSet

```
bool ViewerAutoCenterSet(bool ac)
```

Sets whether or not the scene will automatically be recentered when a change in the scene occurs (e.g. a new object is made active, objects are hidden or shown).

17.4.313 ViewerAutoFitGet

```
bool ViewerAutoFitGet()
```

Returns whether or not the contents of the scene will automatically be fit to the screen when a scene change occurs.

17.4.314 ViewerAutoFitSet

```
bool ViewerAutoFitSet(bool af)
```

Sets whether or not the contents of the scene will automatically be fit to the screen when a scene change occurs.

17.4.315 ViewerBackgroundColorGet

```
OESystem::OEColor ViewerBackgroundColorGet()
```

Returns the background color of the 3D display.

17.4.316 ViewerBackgroundColorSet

```
OESystem::OEColor ViewerBackgroundColorSet(const OESystem::OEColor &color)
```

Sets the background color of the 3D display.

17.4.317 ViewerBookmarkLoad

```
bool ViewerBookmarkLoad(std::string bm)
```

Loads the specified bookmark.

17.4.318 ViewerBookmarksGetAnimated

```
bool ViewerBookmarksGetAnimated()
```

Returns whether or not the transition between bookmarks is animated.

17.4.319 ViewerBookmarksGetAnimationTime

```
unsigned int ViewerBookmarksGetAnimationTime()
```

Returns the default desired time for how long the transition between bookmarks should take if animated.

17.4.320 ViewerBookmarksSetAnimated

```
void ViewerBookmarksSetAnimated(bool animated)
```

Sets whether or not the transition between bookmarks is animated.

17.4.321 ViewerBookmarksSetAnimationTime

```
void ViewerBookmarksSetAnimationTime(unsigned int msecs)
```

Sets the default desired time for how long the transition between bookmarks should take if animated.

17.4.322 ViewerCenterAndRadiusGet

```
std::vector<float> ViewerCenterAndRadiusGet()
```

Returns the current center and radius of the scene in the 3D display in a four-membered list. The first three values in the list are the x,y,z coordinates of the center. The fourth value is the radius.

17.4.323 ViewerCenterAndRadiusSet

```
std::vector<float> ViewerCenterAndRadiusSet(float x, float y, float z,
                                            float radius, bool redraw=true)
```

Sets the current center and radius of the scene in the 3D display. The *redraw* parameter determines whether or not the scene should be redrawn immediately.

17.4.324 ViewerCenterGet

```
std::vector<float> ViewerCenterGet()
```

Returns the current center of the scene as a list of three values corresponding to the x,y,z coordinates. If for some reason the 3D display is invalid, the list returned will be empty.

17.4.325 ViewerCenterSet

```
std::vector<float> ViewerCenterSet(unsigned int, bool redraw=true)
std::vector<float> ViewerCenterSet(float x, float y, float z, bool redraw=true)
```

Sets the current center of the scene. The function that takes three floating point parameters explicitly sets the center of the scene to the coordinates specified by the *x*, *y*, and *z* parameters. The *redraw* parameter specifies whether or not the scene should be immediately redrawn.

The function that takes a single unsigned integer parameter (*id*) attempts to center the scene based on the geometry of the object corresponding to the *id* parameter. If for some reason, a center could not be calculated, this function will return an empty list, otherwise it will return the new center.

17.4.326 ViewerCenterSetScoped

```
std::vector<float>
  ViewerCenterSetScoped(unsigned int scope = BestScope,
                        bool redraw=true)
```

Sets the current center of the scene based on the geometry of the objects found within the specified scope. The *redraw* parameter specifies whether or not the scene should be immediately redrawn. If for some reason, a center could not be calculated, this function will return an empty list, otherwise it will return the new center.

17.4.327 ViewerDepthCueFollowsSlab

```
bool ViewerDepthCueFollowsSlab()
bool ViewerDepthCueFollowsSlab(bool enable)
```

Sets whether or not the depthcue parameters should be adjusted as the slabbing parameters change to maintain the best possible lighting for the scene.

17.4.328 ViewerDepthcueEndGet

```
float ViewerDepthcueEndGet()
```

Returns the spatial end point used in depthcue calculations. Ordinarily, this function should not need to be called by the user.

17.4.329 ViewerDepthcueEndSet

```
float ViewerDepthcueEndSet(float end, bool redraw=true)
```

Sets the spatial end point used in depthcue calculations. Ordinarily, this function should not need to be called by the user.

17.4.330 ViewerDepthcueGet

```
bool ViewerDepthcueGet()
```

Returns whether or not depthcueing is currently enabled in the 3D display.

17.4.331 ViewerDepthcueSet

```
bool ViewerDepthcueSet(bool state)
```

Sets whether or not depthcueing is currently enabled in the 3D display.

17.4.332 ViewerDepthcueStartGet

```
float ViewerDepthcueStartGet()
```

Returns the spatial start point used in depthcue calculations. Ordinarily, this function should not need to be called by the user.

17.4.333 ViewerDepthcueStartSet

```
float ViewerDepthcueStartSet(float start, bool redraw=true)
```

Sets the spatial start point used in depthcue calculations. Ordinarily, this function should not need to be called by the user.

17.4.334 ViewerDiffuseLightGet

```
OESystem::OEColor ViewerDiffuseLightGet()
```

Returns the OpenGL diffuse light property.

17.4.335 ViewerDiffuseLightSet

```
OESystem::OEColor ViewerDiffuseLightSet(const OESystem::OEColor &color)
```

Sets the OpenGL diffuse light property.

17.4.336 ViewerDiffuseMaterialGet

```
OESystem::OEColor ViewerDiffuseMaterialGet()
```

Returns the OpenGL diffuse material property.

17.4.337 ViewerDiffuseMaterialSet

```
OESystem::OEColor ViewerDiffuseMaterialSet(const OESystem::OEColor &color)
```

Set the OpenGL diffuse material property.

17.4.338 ViewerDrawDepictionsGet

```
bool ViewerDrawDepictionsGet()
```

Returns whether or not a 2D depiction is drawn in the 3D display for the active molecule.

17.4.339 ViewerDrawDepictionsSet

```
bool ViewerDrawDepictionsSet(bool show)
```

Sets whether or not a 2D depiction is drawn in the 3D display for the active molecule.

17.4.340 ViewerFit

```
void ViewerFit(float buffer=0.0f)
void ViewerFit(const std::vector<OEPropDB::OEKey> &keys, float buffer=0.0f)
```

This function ensures that the entire contents of the scene fit within the 3D display window. The *buffer* parameter allows specification of an additional buffer applied to the radius of the scene.

The function that takes a list of keys as a parameter will perform the same function as above, but will only ensure that those objects corresponding to the specified keys will be guaranteed to fit within the 3D display window.

17.4.341 ViewerFontSizeGet

```
unsigned int ViewerFontSizeGet()
```

Return the size of the font used for text drawn in the 3D display.

17.4.342 ViewerFontSizeSet

```
void ViewerFontSizeSet(unsigned int sz)
```

Sets the size of the font used for text drawn in the 3D display.

17.4.343 ViewerForwardGet

```
std::vector<float> ViewerForwardGet()
```

Returns the forward facing vector for the OpenGL camera.

17.4.344 ViewerLODGet

```
int ViewerLODGet()
```

Returns the level of display (rendering quality) being used in 3D display window.

17.4.345 ViewerLODSet

```
int ViewerLODSet(unsigned int lod, bool redraw=true)
```

Sets the level of display (rendering quality) being used in the 3D display.

17.4.346 ViewerLightPositionGet

```
std::vector<float> ViewerLightPositionGet()
```

Returns the position of the OpenGL light source.

17.4.347 ViewerLightPositionSet

```
std::vector<float> ViewerLightPositionSet(const std::vector<float> &pos)
```

Sets the position of the OpenGL light source.

17.4.348 ViewerLookAt

```
void ViewerLookAt(const OEPropDB::OEKey &centerKey,
                  const OEPropDB::OEKey &refKey, bool redraw=true)
void ViewerLookAt(const std::vector<float> &center,
                  const std::vector<float> &look,
                  const std::vector<float> &up, bool redraw=true)
```

Sets up the display such that the object corresponding to *centerKey* is at the center of the display with the camera looking down the axis between *centerKey* and the object corresponding to *refKey*.

17.4.349 ViewerMirrorSlabsGet

```
bool ViewerMirrorSlabsGet()
```

Returns whether or not the position of the front and back clipping planes mirror each other.

17.4.350 ViewerMirrorSlabsSet

```
bool ViewerMirrorSlabsSet(bool mirror)
```

Sets whether or not the position of the front and back clipping planes mirror each other.

17.4.351 ViewerNiceFontsGet

```
bool ViewerNiceFontsGet()
```

Returns whether or not text is rendered in the 3D display using high-quality fonts versus simple fonts. Some graphics cards cannot properly handle the high-quality fonts and it is recommended in those situations not to use “nice” fonts.

17.4.352 ViewerNiceFontsSet

```
bool ViewerNiceFontsSet(bool nice)
```

Sets whether or not text is rendered in the 3D display using high-quality fonts versus simple fonts. Some graphics cards cannot properly handle the high-quality fonts and it is recommended in those situations not to use “nice” fonts.

17.4.353 ViewerOrientationGet

```
std::vector<float> ViewerOrientationGet()
```

Returns the OpenGL camera orientation matrix.

17.4.354 ViewerOrientationSet

```
std::vector<float> ViewerOrientationSet(std::vector<float> orientationMatrix)
```

Sets the OpenGL camera orientation matrix.

17.4.355 ViewerProjectorModeGet

```
bool ViewerProjectorModeGet()
```

Returns whether or not the display is being drawn in "Projector Mode". "Projector Mode" is a display mode which is designed to improve visibility when the application is projected using an LCD projector by using a lighter background and a different atom coloring scheme.

17.4.356 ViewerProjectorModeSet

```
void ViewerProjectorModeSet(bool state)
```

Sets whether or not the display is being drawn in "Projector Mode". "Projector Mode" is a display mode which is designed to improve visibility when the application is projected using an LCD projector by using a lighter background and a different atom coloring scheme.

17.4.357 ViewerRadiusGet

```
float ViewerRadiusGet()
```

Returns the current radius of the scene in the 3D display window.

17.4.358 ViewerRadiusSet

```
float ViewerRadiusSet(float radius, bool redraw=true)
```

Sets the current radius of the scene in the 3D display window. The *redraw* parameter specifies whether or not to immediately redraw the scene.

17.4.359 ViewerRecenter

```
void ViewerRecenter()
```

This function recenters the scene based on the geometry of the contents of the scene.

17.4.360 ViewerReprobeStereo

```
bool ViewerReprobeStereo()
```

This function rechecks to see whether hardware stereo is supported on the current machine. Ordinarily, this function should not need to be called by the user.

17.4.361 ViewerRotate

```
void ViewerRotate(const std::string &axis, float angle, bool redraw=true)
```

This function rotates the display around the specified axis (valid parameters: 'x','y','z' for screen or trackball axis, 'X', 'Y', 'Z' for world axis) by *angle* degrees. The center of rotation is the world center. The *redraw* parameter specifies whether or not to immediately redraw the scene.

17.4.362 ViewerScaleGet

```
float ViewerScaleGet()
```

Returns the scale of the scene in the 3D display window.

17.4.363 ViewerScaleSet

```
float ViewerScaleSet(float radius, bool redraw=true)
```

Sets the scale of the scene in the 3D display window.

17.4.364 ViewerShininessMaterialGet

```
float ViewerShininessMaterialGet()
```

Returns the OpenGL material shininess property.

17.4.365 ViewerShininessMaterialSet

```
float ViewerShininessMaterialSet(float shine)
```

Sets the OpenGL material shininess property.

17.4.366 ViewerShowActiveBorderGet

```
bool ViewerShowActiveBorderGet()
```

Returns whether or not a blue border is drawn around the active pane in multi-pane display mode.

17.4.367 ViewerShowActiveBorderSet

```
void ViewerShowActiveBorderSet(bool show)
```

Sets whether or not a blue border is drawn around the active pane in multi-pane display mode.

17.4.368 ViewerShowGridGet

```
bool ViewerShowGridGet()
```

Returns whether or not borders are drawn around individual panes in multi-pane display mode.

17.4.369 ViewerShowGridSet

```
void ViewerShowGridSet(bool show)
```

Sets whether or not borders are drawn around individual panes in multi-pane display mode.

17.4.370 ViewerShowTrackballGuideSet

```
void ViewerShowTrackballGuideSet(bool show)
```

Sets whether or not a trackball guide will be displayed on the 3D display window to help guide what is considered inside and outside with respect to mouse actions. Please see ViewerMouseOutsideAwareGet/Set functions for more details.

17.4.371 ViewerSlabEnableGet

```
bool ViewerSlabEnableGet()
```

Returns whether or not slabbing (clipping) of the 3D display is enabled.

17.4.372 ViewerSlabEnableSet

```
bool ViewerSlabEnableSet(bool enable)
```

Sets whether or not slabbing (clipping) of the 3D display is enabled. The *redraw* parameter specifies whether or not the scene should be immediately redrawn.

17.4.373 ViewerSlabFarGet

```
float ViewerSlabFarGet()
```

Returns the spatial position of the far clipping plane. Ordinarily, this function should not need to be called by the user.

17.4.374 ViewerSlabFarSet

```
float ViewerSlabFarSet(float radius, bool redraw=true)
```

Sets the spatial position of the far clipping plane. The *redraw* parameter specifies whether or not the scene should be immediately redrawn. Ordinarily, this function should not need to be called by the user.

17.4.375 ViewerSlabNearGet

```
float ViewerSlabNearGet()
```

Returns the spatial position of the near clipping plane. Ordinarily, this function should not need to be called by the user.

17.4.376 ViewerSlabNearSet

```
float ViewerSlabNearSet(float radius, bool redraw=true)
```

Sets the spatial position of the near clipping plane. The *redraw* parameter specifies whether or not the scene should be immediately redrawn. Ordinarily, this function should not need to be called by the user.

17.4.377 ViewerSlabWidthGet

```
float ViewerSlabWidthGet()
```

Returns the distance between the near and far clipping planes. Ordinarily, this function should not need to be called by the user.

17.4.378 ViewerSlabWidthSet

```
float ViewerSlabWidthSet(float width, bool redraw=true)
```

Sets the distance between the near and far clipping planes. The *redraw* parameter specifies whether or not the scene should be immediately redrawn. Ordinarily, this function should not need to be called by the user.

17.4.379 ViewerSpecularMaterialGet

```
OESystem::OEColor ViewerSpecularMaterialGet()
```

Returns the OpenGL specular material property.

17.4.380 ViewerSpecularMaterialSet

```
OESystem::OEColor ViewerSpecularMaterialSet(const OESystem::OEColor &color)
```

Sets the OpenGL specular material property.

17.4.381 ViewerStereoAngleGet

```
float ViewerStereoAngleGet()
```

Returns the angle used when calculating stereographic display offsets.

17.4.382 ViewerStereoAngleSet

```
float ViewerStereoAngleSet(float angle)
```

Sets the angle used when calculating stereographic display offsets.

17.4.383 ViewerStereoCrossEyedGet

```
bool ViewerStereoCrossEyedGet()
```

Returns whether or not the stereographic display is in cross-eyed mode.

17.4.384 ViewerStereoCrossEyedSet

```
bool ViewerStereoCrossEyedSet(bool cross)
```

Sets whether or not the stereographic display is in cross-eyed mode.

17.4.385 ViewerStereoEnableGet

```
bool ViewerStereoEnableGet()
```

Returns whether or not stereographic display is enabled.

17.4.386 ViewerStereoEnableSet

```
bool ViewerStereoEnableSet(bool enabled)
```

Sets whether or not stereographic display is enabled.

17.4.387 ViewerStereoSeparationGet

```
float ViewerStereoSeparationGet()
```

Returns the eye separation used in stereographic display mode.

17.4.388 ViewerStereoSeparationSet

```
float ViewerStereoSeparationSet(float separation)
```

Sets the eye separation used in stereographic display mode.

17.4.389 ViewerStereoStyleGet

```
unsigned char ViewerStereoStyleGet()
```

Sets the stereographic display mode (hardware, splitscreen, or none).

17.4.390 ViewerStereoStyleSet

```
unsigned char ViewerStereoStyleSet(unsigned char style)
```

Sets the stereographic display mode (hardware, stencil, splitscreen, or none).

17.4.391 ViewerStyleControlVisibleGet

```
bool ViewerStyleControlVisibleGet(const std::string &style)
```

Returns whether or not the specified style control widget is currently visible above the 3D display. Available style widgets include: “color”, “selection”, “style”, “contours”, and “graphics”.

17.4.392 ViewerStyleControlVisibleSet

```
void ViewerStyleControlVisibleSet(const std::string &style, bool vis)
```

Sets whether or not the specified style control widget is currently visible above the 3D display. Available style widgets include: “color”, “selection”, “style”, “contours”, and “graphics”.

17.4.393 ViewerSupportsHWStereo

```
bool ViewerSupportsHWStereo()
```

Returns whether or not the machine the application is being run on supports the use hardware stereo.

17.4.394 ViewerTextFontGet

```
std::string ViewerTextFontGet()
```

Returns the font currently being used when rendering text in the 3D display. This font only applies when using "nice" fonts.

17.4.395 ViewerTextFontSet

```
void ViewerTextFontSet(const std::string &)
```

Sets the font currently being used when rendering text in the 3D display. This font only applies when using "nice" fonts.

17.4.396 ViewerTextScaleGet

```
float ViewerTextScaleGet()
```

Returns the current global scale for text drawn in the 3D display window.

17.4.397 ViewerTextScaleSet

```
float ViewerTextScaleSet(float scale, bool redraw=true)
```

Sets the current global scale for text drawn in the 3D display window.

17.4.398 ViewerToggleRenderFeatures

```
void ViewerToggleRenderFeatures(unsigned int features, bool on)
```

Toggles the use of certain advanced visualization features.

17.4.399 ViewerTranslateX

```
void ViewerTranslateX(float val)
```

Translates the display along the X axis by the specified amount (in Angstroms).

17.4.400 ViewerTranslateY

```
void ViewerTranslateY(float val)
```

Translates the display along the Y axis by the specified amount (in Angstroms).

17.4.401 ViewerTranslateZ

```
void ViewerTranslateZ(float val)
```

Translates the display along the Z axis by the specified amount (in Angstroms).

17.4.402 ViewerUpGet

```
std::vector<float> ViewerUpGet()
```

Returns the OpenGL camera up vector.

17.4.403 ViewerUseDisplayListGet

```
bool ViewerUseDisplayListGet()
```

Returns whether or not the 3D renderer uses a display list to encapsulate the master scene. The default is false. Ordinarily, this function does not need to be called by the user.

17.4.404 ViewerUseDisplayListSet

```
bool ViewerUseDisplayListSet(bool use)
```

Sets whether or not the 3D renderer uses a display list to encapsulate the master scene. The default is false. Ordinarily, this function does not need to be called by the user. Setting this property to true may have cause significant negative performance on certain machines.

17.4.405 ViewerUseSystemFontsGet

```
bool ViewerUseSystemFontsGet()
```

Returns whether or not the 3D renderer uses system fonts when displaying text.

17.4.406 ViewerUseSystemFontsSet

```
bool ViewerUseSystemFontsSet(bool state)
```

Sets whether or not the 3D renderer uses system fonts when display text.

17.4.407 ViewerSetShowObjectToolbar

```
void ViewerSetShowObjectToolbar(bool state)
```

Sets whether or not the style & color buttons are shown in the bottom toolbar for the active object

17.4.408 ViewerGetShowObjectToolbar

```
bool ViewerGetShowObjectToolbar()
```

Sets whether or not the style & color buttons are shown in the bottom toolbar for the active object

17.5 Object Functions

The functions detailed in this section provide a mechanism to add, delete, modify, and organize the many types of objects (e.g. molecules, grids, and surfaces) available in VIDA.

17.5.1 Add

```
unsigned int Add(OESystem::OEBase &, unsigned int listID = 0)
```

Adds the specified object to the application and returns a unique ID associated with that object. If the optional listID argument is provided, then the object is added to the specified list. Otherwise, a new list is created and the object is added to it.

17.5.2 AddCSVSmiles

```
unsigned int AddCSVSmiles( const std::string &smiles,
                           const std::vector<std::string> &headers,
                           const std::vector<std::string> &values,
                           const std::string &filename = "",
                           unsigned int       listid   = 0)
```

Adds a new molecule from SMILES with the optional associated data. *smiles* species the SMILES string to parse, *headers* is a list of column names corresponding to the data specified in *values*, *filename* is an optional filename to attribute this molecule to, and *listid* is the ID of the list to which this molecule will be added. If *listid* is zero, a new list will be created.

Returns the ID of the list to which the molecule was added.

17.5.3 AddURLMol

```
unsigned int AddURLMol( const std::string              &url,
                        const std::string              &urlFunc,
                        const std::vector<std::string> &headers,
                        const std::vector<std::string> &values,
                        const std::string              &filename,
                        unsigned int                    listid)
```

Reserved for future implementation.

17.5.4 AtomDataGet

```
AtomData AtomDataGet(const OEPropDB::OEKey &key )
```

Returns an AtomData object populated with information about the atom specified by the parameter *key*.

17.5.5 CheckIn

```
bool CheckIn(OESystem::OEBase &)
```

Checks the specified Python accessible object back into the main VIDA application which will synchronize any changes made to the object in Python with the object in VIDA. The object in question must have been checked out of VIDA prior to being checked back in.

See *GridCheckOut*, *MoleculeCheckOut*, and *SurfaceCheckOut*.

17.5.6 ChildrenGet

```
std::vector<OEPropDB::OEKey> ChildrenGet(unsigned int id)
std::vector<OEPropDB::OEKey> ChildrenGet(const OEPropDB::OEKey &)
```

Returns a list of keys corresponding to any child objects associated with the specified object.

17.5.7 ContourCloudJitterDefaultGet

```
float ContourCloudJitterDefaultGet()
```

Returns the value (between 0.0 and 1.0) indicated by the Jitter slider on the Contours panel.

17.5.8 ContourCloudJitterGet

```
float ContourCloudJitterGet(unsigned int id)
```

Returns the jitter value assigned to grid *id*. The jitter value is used when the contour is displayed with the "cloud" style, and represents the degree to which the grid vertex positions are to be randomized. Jitter values range from 0.0 (no jitter) to 1.0 (jitter equal to the grid spacing).

17.5.9 ContourCloudJitterSet

```
float ContourCloudJitterSet(unsigned int id, float jitter)
```

Sets the jitter value to be used when grid *id* is displayed with the "cloud" style. Jitter values range from 0.0 (no jitter) to 1.0 (jitter equal to the grid spacing).

17.5.10 ContourCountScoped

```
int ContourCountScoped(unsigned int scope = BestScope)
```

Returns the number of contours for all grids in *scope*.

This returns the number of contours for the focused grid or the first grid found in the given scope.

17.5.11 ContourCreate

```
void ContourCreate(unsigned int id=0)
void ContourCreate(unsigned int id, float threshold)
void ContourCreate(unsigned int id, float threshold,
                   const OESystem::OEColor &color)
```

Create a new contour for grid *id*.

17.5.12 ContourCreateSurface

```
unsigned int ContourCreateSurface(unsigned int id, unsigned int contourIndex)
```

Create a contour surface for the surface defined by the repository id *id* for the contour referenced by *contourIndex*.

17.5.13 ContourDeleteAll

```
void ContourDeleteAll(unsigned int id)
```

Delete all contours for grid *id*.

17.5.14 ContourDeleteByIndex

```
void ContourDeleteByIndex(unsigned int id, unsigned int index)
```

Delete contour *index* for grid *id*. All contours with indices greater than *index* will now be at their current index - 1.

17.5.15 ContourDeleteByThreshold

```
void ContourDeleteByThreshold(unsigned int id, float threshold)
```

Delete all contours above *threshold* for grid *id*. Contour indices will be reassigned.

17.5.16 ContourExtractIsoSurface

```
unsigned int ContourExtractIsoSurface(unsigned int id, unsigned int vertex,
                                      bool maskVis=true)
```

Reserved for internal use.

17.5.17 ContourFixAsSurfaceScoped

```
std::vector<unsigned int>
  ContourFixAsSurfaceScoped(unsigned int scope = BestScope)
```

Creates an actual surface object for each contour in the specified scope. Returns a list of IDs corresponding to the generated surfaces.

17.5.18 ContourGetAll

```
std::vector<float> ContourGetAll(unsigned int id=0)
```

Get all contour levels from each contour drawn for grid *id*.

17.5.19 ContourLevelForIndexGet

```
float ContourLevelForIndexGet(unsigned int id, unsigned int index)
```

Returns the contour level for grid *id* at index *id*.

17.5.20 ContourLevelForIndexGetScoped

```
float ContourLevelForIndexGetScoped(int index,
                                    unsigned int scope = BestScope)
```

Returns the contour level for index *id* for the first grid in *scope*.

17.5.21 ContourLevelForIndexSet

```
float ContourLevelForIndexSet(unsigned int id, unsigned int index, float thresh)
```

Set the level for the contour at index *index* to *thresh* for grid *id*.

17.5.22 ContourLevelForIndexSetScoped

```
void ContourLevelForIndexSetScoped(unsigned int index, float thresh,
                                   unsigned int gridType=UINT_MAX,
                                   unsigned int scope = BestScope)
```

Set the level for the contour at index *index* to *thresh* for all grids in *scope*.

17.5.23 ContourLevelNudgeScoped

```
void ContourLevelNudgeScoped(bool increase,
                             unsigned int scope = BestScope)
```

This function increases or decreases the contour value for the contours in the specified scope by 0.1 depending on whether the *increase* parameter is set to true or false.

17.5.24 ContourLevelSetScoped

```
void ContourLevelSetScoped(float level,
                           unsigned int scope = BestScope)
```

This function sets the contour level of all the contours in the specified scope to the value specified by the parameter *level*.

17.5.25 ContourMax

```
float ContourMax(unsigned int oerid)
```

Returns the maximum allowable contour threshold for grid *id*.

17.5.26 ContourMaxScoped

```
float ContourMaxScoped(unsigned int scope = BestScope)
```

Returns the maximum allowable contour threshold for the first grid found in *scope*.

17.5.27 ContourMin

```
float ContourMin(unsigned int oerid)
```

Returns the minimum allowable contour threshold for grid *id*.

17.5.28 ContourMinScoped

```
float ContourMinScoped(unsigned int scope = BestScope)
```

Returns the minimum allowable contour threshold for the first grid found in *scope*.

17.5.29 ContourRadiusGet

```
float ContourRadiusGet(unsigned int id=0)
```

Returns the radius used when generating contours for crystallographic grids.

17.5.30 ContourRadiusSet

```
float ContourRadiusSet(unsigned int id, float rad)
```

Sets the radius used when generating contours for crystallographic grids.

17.5.31 ContourResolutionGet

```
float ContourResolutionGet(unsigned int id=0)
```

Return the contouring resolution for drawing the contours in Angstroms. A lower resolution samples more grid points, but requires more memory.

17.5.32 ContourResolutionSet

```
float ContourResolutionSet(unsigned int id, float rad)
```

Set the contouring resolution in Angstroms. Lower values sample more grid points and generate finer visualizations but require more memory.

17.5.33 ContourTypedAddScoped

```
void ContourTypedAddScoped(unsigned int scope = BestScope)
```

Add a single contour to all grids of type *type* in *scope*.

17.5.34 ContourTypedCountScoped

```
int ContourTypedCountScoped(unsigned int scope = BestScope)
```

Return the number of contours of all grids of type *type* in *scope*. This essentially returns the number of contours of the first grid found.

17.5.35 ContourTypedMaxScoped

```
float ContourTypedMaxScoped(unsigned int scope = BestScope)
```

Returns the maximum allowable threshold for the first grid found of type *type* in *scope*.

17.5.36 ContourTypedMinScoped

```
float ContourTypedMinScoped(unsigned int scope = BestScope)
```

Returns the minimum allowable threshold for the first grid found of type *type* in *scope*.

17.5.37 ContourTypedRemoveScoped

```
void ContourTypedRemoveScoped(unsigned int scope = BestScope)
```

Remove the last contour from all grids of type *type* in *scope*.

17.5.38 ContourTypedSetLevelForIndexScoped

```
void ContourTypedSetLevelForIndexScoped(unsigned int index, float thresh,
                                        unsigned int scope = BestScope)
```

Set the contour *level* for the contour at *index* for all grids of type *type* in *scope*.

17.5.39 ContourVolume

```
float ContourVolume(unsigned int id, unsigned int contourIndex)
```

Return the enclosed volume of Grid *id*'s contour at index *contourIndex*.

17.5.40 Delete

```
void Delete(unsigned int id)
void Delete(const std::vector<unsigned int> &)
void Delete(const std::vector<OEPropDB::OEKey> &)
void Delete(const OEPropDB::OEKey &, bool all=false)
```

Deletes the specified object(s) from the current session. It is important to note that this does NOT delete the actual files (or entries within files) associated with the specified objects.

17.5.41 DeleteAll

```
void DeleteAll(bool force=false)
```

Clears the application back to a clean state with nothing loaded.

17.5.42 DeleteScoped

```
void DeleteScoped(unsigned int scope, bool inScope)
```

Scoped version of *Delete* but with the additional option to specify whether to delete everything within the scope or everything not within the scope using the *inScope* parameter.

17.5.43 FindByDataScoped

```
std::vector<OEPropDB::OEKey> FindByDataScoped(const std::string &query,
                                              unsigned int scope = AllScope)
```

Reserved for future implementation.

17.5.44 FindByQueryScoped

```
std::vector<OEPropDB::OEKey> FindByQueryScoped(const OEPropDB::OEKey &query,
                                               unsigned int scope = AllScope)
std::vector<OEPropDB::OEKey> FindByQueryScoped(const OEPropDB::OEKey &query,
                                               unsigned int atoms,
                                               unsigned int bonds,
                                               unsigned int scope = AllScope)
```

Uses the specified molecule as a query to search the specified scope. Returns a list of keys corresponding to molecules that match the specified query.

17.5.45 FindBySMARTSScoped

```
std::vector<OEPropDB::OEKey> FindBySMARTSScoped(const std::string &smarts,
                                                unsigned int scope = AllScope)
```

Uses the specified SMARTS pattern as a query to search the specified scope. Returns a list of keys corresponding to molecules that match the specified SMARTS pattern.

17.5.46 FindBySimilarityScoped

```
std::vector<OEPropDB::OEKey> FindBySimilarityScoped(const OEPropDB::OEKey &query,
                                                    float similarity,
                                                    unsigned int scope = AllScope)
```

Searches the specified scope for molecules with a Lingo similarity greater than the specified *similarity* cutoff with the specified molecule *query*.

17.5.47 FindByTitleScoped

```
std::vector<OEPropDB::OEKey> FindByTitleScoped(const std::string &expr,
                                               unsigned int scope = AllScope)
```

Uses the specified title query as a regular expression to search the specified scope for molecules whose title match the expression.

17.5.48 FindInRepository

```
void FindInRepository()
```

This function prompts the user to specify a query to search the contents of the application repository. The user then has the option to select some, none, or all of the hits to be placed in a new list.

17.5.49 FindOnDisk

```
void FindOnDisk()
```

This function prompts the user to specify a query to search a specific region of the user's hard drive for molecules matching the query. The user then has the option to load some, none, or all of the files containing the hits. This function will load the entire file where a hit was discovered as opposed to just the hits in that file.

17.5.50 GridAdd

```
unsigned int GridAdd(OESystem::OESkewGrid &grd, unsigned int listID = 0)
unsigned int GridAdd(OESystem::OEScalarGrid &grd, unsigned int listID = 0)
```

Adds the specified Python grid object to VIDA. If the optional listID argument is provided, then the grid is added to the specified list. Otherwise, a new list is created and the grid is added to it.

17.5.51 GridCheckIn

```
bool GridCheckIn(OESystem::OESkewGrid &grd,
                 unsigned int checkInType = OECheckInType_UnknownChange)
bool GridCheckIn(OESystem::OEScalarGrid &grd,
                 unsigned int checkInType = OECheckInType_UnknownChange)
```

Checks the specified Python accessible grid back into the main VIDA application which will synchronize any changes made to the grid in Python with the grid in VIDA. The grid in question must have been checked out of VIDA prior to being checked back in.

The *checkInType* parameter tells VIDA what type of changes were made to the grid in order to optimize the check in process. There are multiple allowable values for this parameter which can be OR'd together if necessary:

- OECheckInType_ConformerChange
- OECheckInType_CoordinateChange
- OECheckInType_DataChange
- OECheckInType_GraphChange
- OECheckInType_NoChange
- OECheckInType_RenderChange
- OECheckInType_AnnotationChange
- OECheckInType_UnknownChange

17.5.52 GridCheckOut

```
bool GridCheckOut(OESystem::OESkewGrid &grd, unsigned int id)
bool GridCheckOut(OESystem::OEScalarGrid &grd, unsigned int id)
```

Checks out a copy of the grid specified by *id* into the specified Python grid object (*grd*). Returns whether or not the check out process succeeded.

The checked out grid can be modified in Python, but those changes will not be applied to the original grid in VIDA until the modified grid is checked back in using the *GridCheckIn* command.

17.5.53 GridClear

```
void GridClear(unsigned int id)
```

Clears the data on the specified grid.

17.5.54 GridCopy

```
unsigned int GridCopy(unsigned int id)
```

Creates a copy of the specified grid and returns the ID of the newly created copy.

17.5.55 GridCreateElectrostaticsGrid

```
unsigned int GridCreateElectrostaticsGrid(OEPropDB::OEKey k,
                                          bool property=false)
```

Create an electrostatics grid for molecule *id*.

If *property* is true, then this grid is attached to *id* and does not show up in the list window. These grids cannot be saved to disk.

If *property* is false, then this grid is created as a separate entity from *id*.

17.5.56 GridCreateGaussian

```
unsigned int GridCreateGaussian(const OEPropDB::OEKey &k, float res)
```

Creates a gaussian grid representation for the molecule associated with the specified key at the specified resolution.

17.5.57 GridCreateGaussianProduct

```
unsigned int GridCreateGaussianProduct(const OEPropDB::OEKey &k, float res)
```

Creates a gaussian product grid representation for the molecule associated with the specified key at the specified resolution.

17.5.58 GridDefaultContourLevelByIndexGet

```
float GridDefaultContourLevelByIndexGet(unsigned int gridType,
                                        unsigned int index)
```

Returns the default contour level for contour *index* for grids of type *gridType*.

17.5.59 GridDefaultContourLevelByIndexSet

```
void GridDefaultContourLevelByIndexSet(unsigned int gridType,
                                       unsigned int index, float thresh)
```

Set the default contour level *thresh* for contour *index* for grids of type *gridType*.

17.5.60 GridDefaultNumContoursGet

```
unsigned int GridDefaultNumContoursGet(unsigned int gridType)
```

Returns the default number of contours for grids of type *gridType*.

17.5.61 GridDefaultNumContoursSet

```
void GridDefaultNumContoursSet(unsigned int gridType, unsigned int count)
```

Set the default number of contours for grids of type *gridType* to *count*.

17.5.62 GridInitializeContours

```
void GridInitializeContours(unsigned int id, bool update=true)
```

Initialize a grid to its default contours, colors and levels.

17.5.63 GridNormalize

```
bool GridNormalize(unsigned int id)
```

Normalizes a grid. This finds the sigma for the grid and divides all grid points by that value.

17.5.64 GridRegularize

```
unsigned int GridRegularize(unsigned int id)
```

Regularizes the specified skew grid. Returns the ID of the newly created regular scalar grid.

17.5.65 GridToGaussianGrid

```
unsigned int GridToGaussianGrid(unsigned int id)
```

Creates a gaussian grid representation for the specified grid.

17.5.66 GridTypeGet

```
unsigned int GridTypeGet(unsigned int oerid)
```

Returns the type of grid for grid *id*.

17.5.67 GridTypeGetScoped

```
unsigned int GridTypeGetScoped(unsigned int scope = BestScope)
```

Returns the type of grid for the first grid found in *scope*.

17.5.68 GridTypeSet

```
void GridTypeSet(unsigned int id, unsigned int gridType)
```

Set the grid type of grid *id* to *gridType*.

17.5.69 GridTypeSetScoped

```
void GridTypeSetScoped(unsigned int gridType,
                       unsigned int scope = BestScope)
```

Scoped version of GridTypeSet.

17.5.70 GridWorkingGetScoped

```
unsigned int GridWorkingGetScoped(unsigned int scope = BestScope)
```

Returns the ID for the "working grid" which is determined in the following order: grid associated with selected contour, grid which is currently focused, grid which is a child object of the currently focused object, or lastly the first grid found in the current default scope.

17.5.71 HasGridChildrenScoped

```
int HasGridChildrenScoped(unsigned int scope = BestScope)
```

Returns whether or not any of the molecules in the specified scope have visible grids attached to them. Returns 0 for none of the molecules have visible grids, 1 for all molecules have visible grid, and 2 for at least one has a visible grid.

17.5.72 HasSurfaceChildrenScoped

```
int HasSurfaceChildrenScoped(const std::string &type,
                             unsigned int scope = BestScope)
```

Returns whether or not any of the molecules in the specified scope have visible surfaces attached to them. Returns 0 for none of the molecules have visible surfaces, 1 for all molecules have visible surface, and 2 for at least one has a visible surface.

17.5.73 Initialize

```
void Initialize(unsigned int id)
void Initialize(const OEPropDB::OEKey &)
```

Initializes the specified object for display. This function is automatically called on objects that are loaded into VIDA and ordinarily should not need to be called by the user.

17.5.74 IsAGrid

```
bool IsAGrid(unsigned int id)
bool IsAGrid(const OEPropDB::OEKey &k)
```

Returns whether or not the specified object is a grid.

17.5.75 IsAList

```
bool IsAList(unsigned int id)
```

Returns whether or not the specified object is a list.

17.5.76 IsAMolecule

```
bool IsAMolecule(unsigned int id)
bool IsAMolecule(const OEPropDB::OEKey &k)
```

Returns whether or not the specified object is a molecule.

17.5.77 IsAReflection

```
bool IsAReflection(unsigned int id)
bool IsAReflection(const OEPropDB::OEKey &k)
```

Returns whether or not the specified object is a reflection.

17.5.78 IsASmallMolecule

```
bool IsASmallMolecule(unsigned int id)
bool IsASmallMolecule(const OEPropDB::OEKey &k)
```

Returns whether or not the specified object is a small molecule.

17.5.79 IsASurface

```
bool IsASurface(unsigned int id)
bool IsASurface(const OEPropDB::OEKey &k)
```

Returns whether or not the specified object is a surface.

17.5.80 KeyGet

```
OEPropDB::OEKey KeyGet(const OESystem::OEBase &)
OEPropDB::OEKey KeyGet(const OESystem::OEBase &, const OEPropDB::OEKey &pkey)
```

Returns the key associated with the specified OEBase object. If the object's parent key is known, that can be passed as a parameter as well to aid the key determination process.

17.5.81 KeyIDGet

```
unsigned int KeyIDGet(const OEPropDB::OEKey &)
```

Returns the ID associated with the specified key.

17.5.82 KeyParentIDGet

```
unsigned int KeyParentIDGet(const OEPropDB::OEKey &)
```

Returns the ID of the parent object associated with the specified key. This function should be used for child object types such as atoms, bonds, triangles, and vertices.

17.5.83 KeySourceIDGet

```
unsigned int KeySourceIDGet(const OEPropDB::OEKey &)
```

Returns the ID of the parent object associated with a specified key. This function should be used for top-level object types such as molecules, grids, and surfaces in situations where these objects have been added as children to another object.

17.5.84 KeyTypeGet

```
std::string KeyTypeGet(const OEPropDB::OEKey &)
```

Returns a string representation of the object type associated with the specified key. Return values include:

- "Atom"
- "Bond"
- "Grid"
- "Mol"
- "Surface"
- "Triangle"
- "Vertex"

17.5.85 KeysGet

```
std::vector<OEPropDB::OEKey> KeysGet(unsigned int id)
```

Returns a list of all the keys for the specified ID. For most objects, the list returned will contain a single entry; however, for multi-conformer molecules, the list will include a key for each individual conformer.

17.5.86 ListAddObject

```
bool ListAddObject(unsigned int listid, unsigned int objectid)
```

Adds the object specified by *objectid* to the list specified by *listid*.

17.5.87 ListAddObjects

```
bool ListAddObjects(unsigned int listid,
                    const std::vector<unsigned int> &ids)
```

Adds the objects specified by the list of IDs (*ids*) to the list specified by *listid*.

17.5.88 ListGetNames

```
std::vector<std::string> ListGetNames()
```

Returns a list of the names of all the current lists.

17.5.89 ListGetObjectLists

```
std::vector<unsigned int> ListGetObjectLists(unsigned int objectid)
```

Returns a list of all the lists to which the object specified by *objectid* belongs.

17.5.90 ListGetObjects

```
std::vector<unsigned int> ListGetObjects(unsigned int listid)
```

Returns a list of IDs of all the objects contained within the list specified by *listid*.

17.5.91 ListMoveObject

```
bool ListMoveObject(unsigned int srclist, unsigned int dstlist,
                    unsigned int objectid)
```

Moves the object specified by *objectid* from the list specified by *srclist* to the list specified by *dstlist*.

17.5.92 ListMoveObjects

```
bool ListMoveObjects(unsigned int srclist, unsigned int dstlist,
                     const std::vector<unsigned int> &ids)
```

Moves the objects specified by list of IDs (*ids*) from the list specified by *srclist* to the list specified by *dstlist*.

17.5.93 ListNew

```
unsigned int ListNew(const std::string &name)
unsigned int ListNew(const std::string &name,
                     const std::vector<unsigned int> &ids)
unsigned int ListNew(const std::string &name,
                     const std::vector<OEPropDB::OEKey> &keys)
```

Creates a new list with the specified name. If a list of IDs or keys is specified with this call, all of the objects corresponding to those IDs or keys will be added to the list.

Returns the ID of the created list.

17.5.94 ListNewAnd

```
unsigned int ListNewAnd(const std::vector<unsigned int> &ids)
```

Creates a new list containing all of the objects that each appear in every list specified by the list of IDs (*ids*).

Returns the ID of the created list.

17.5.95 ListNewMarked

```
unsigned int ListNewMarked(const std::string &name="")
```

Creates a new list containing all the currently marked objects. Objects will not be removed from this list if they are subsequently unmarked.

Returns the ID of the created list.

17.5.96 ListNewOr

```
unsigned int ListNewOr(const std::vector<unsigned int> &ids)
```

Creates a new list containing all of the objects in all of the lists specified by the list of IDs (*ids*).

Returns the ID of the created list.

17.5.97 ListNewXor

```
unsigned int ListNewXor(const std::vector<unsigned int> &ids)
```

Creates a new list containing all of the objects that each appear only once among all the lists specified by the list of IDs (*ids*).

Returns the ID of the created list.

17.5.98 ListRemoveObject

```
bool ListRemoveObject(unsigned int listid, unsigned int objectid)
```

Removes the object specified by *objectid* from the list specified by *listid*.

17.5.99 ListRemoveObjects

```
bool ListRemoveObjects( unsigned int listid,
                        const std::vector<unsigned int> &ids )
```

Removes the object(s) specified by the list of IDs (*ids*) from the list specified by *listid*.

17.5.100 ListRootList

```
unsigned int ListRootList()
```

Returns the ID of the root list.

17.5.101 ListSubsetMarked

```
unsigned int ListSubsetMarked(unsigned int id, bool match)
```

Creates a new list containing a subset of the objects contained within the list specified by *id*. If the *match* parameter is True, the subset will contain all of the marked objects in the source list, if *match* is False, the subset will contain all of the non-marked objects in the source list.

Returns the ID of the created list.

17.5.102 ListSubsetQuery

```
unsigned int ListSubsetQuery(unsigned int id, const std::string &, bool match)
```

Creates a new list containing a subset of the objects contained within the list specified by *id*. If the *match* parameter is True, the subset will contain all of the objects that match the specified query in the source list, if *match* is Fale, the subset will contain all of the objects which fail to match the specified query in the source list. The query parameter can be specified as a SMARTS, a SMILES, or an IUPAC name.

Returns the ID of the created list.

17.5.103 MoleculeAdd

```
unsigned int MoleculeAdd(const std::string &, unsigned int listID = 0)
unsigned int MoleculeAdd(OEChem::OEMCMolBase &, unsigned int listID  = 0)
```

Adds a new molecule to VIDA. There are two implementations of this function. The first implementation expects a hexadecimal encoded OEB string specifying a molecule. The second implementation expects a Python molecule object.

If the optional listID argument is provided, then the molecule is added to the specified list. Otherwise, a new list is created and the molecule is added to it.

Returns the ID assigned to the newly added molecule or zero if the add failed for any reason.

17.5.104 MoleculeCheckIn

```
bool MoleculeCheckIn(OEChem::OEMCMolBase &mol,
                     unsigned int checkInType = OECheckInType_UnknownChange )
```

Checks the specified Python accessible molecule back into the main VIDA application which will synchronize any changes made to the molecule in Python with the molecule in VIDA. The molecule in question must have been checked out of VIDA prior to being checked back in.

The *checkInType* parameter tells VIDA what type of changes were made to the molecule in order to optimize the check in process. There are multiple allowable values for this parameter which can be OR'd together if necessary:

- OECheckInType_ConformerChange
- OECheckInType_CoordinateChange
- OECheckInType_DataChange
- OECheckInType_GraphChange
- OECheckInType_NoChange
- OECheckInType_RenderChange
- OECheckInType_AnnotationChange
- OECheckInType_UnknownChange

17.5.105 MoleculeCheckOut

```
bool MoleculeCheckOut(OEChem::OEMCMolBase &mol, unsigned int id)
```

Checks out a copy of the molecule specified by *id* into the specified Python molecule object (*mol*). Returns whether or not the check out process succeeded.

The checked out molecule can be modified in Python, but those changes will not be applied to the original molecule in VIDA until the modified molecule is checked back in using the *MoleculeCheckIn* command.

17.5.106 MoleculeComponentNamesGet

```
std::vector<std::string> MoleculeComponentNamesGet(unsigned int id)
```

Returns a list of names of the individual components of the molecule specified by `id` based on PDB related subsets.

17.5.107 MoleculeExamine

```
bool MoleculeExamine(OEChem::OEMCMolBase &, unsigned int id)
bool MoleculeExamine(OEChem::OEMCMolBase &, const OEPropDB::OEKey &key)
```

Copies the molecule associated with the specified ID or key into the specified Python molecule object. Returns True if the copy happened successfully. In cases where a script needs read-only access to a molecule, MoleculeExamine provides a much faster way to retrieve molecules than MoleculeCheckOut or MoleculeGet. Molecules retrieved using MoleculeExamine cannot be checked back in using MoleculeCheckIn.

17.5.108 MoleculeGet

```
bool MoleculeGet(OEChem::OEMCMolBase &, unsigned int id)
bool MoleculeGet(OEChem::OEMCMolBase &, const OEPropDB::OEKey &key)
OEChem::OEMol MoleculeGet(unsigned int id)
OEChem::OEMol MoleculeGet(const OEPropDB::OEKey &key)
```

Copies the molecule associated with the specified ID or key into the specified Python molecule object. Returns whether or not the copy happened successfully.

17.5.109 MoleculeHasComponents

```
bool MoleculeHasComponents(unsigned int id)
```

Returns whether or not the molecule specified by `id` contains multiple distinct components.

17.5.110 MoleculeMaxResidueGet

```
int MoleculeMaxResidueGet(unsigned int id)
```

Returns the largest residue index of all the residues contained within the specified molecule.

17.5.111 MoleculeMergeScoped

```
unsigned int MoleculeMergeScoped(unsigned int scope = BestScope,
                                 const std::string &name="Merged",
                                 unsigned int listid=0)
```

Merges all of the molecules in the specified scope into a single new molecule. The `name` parameter specifies the new molecule's name and the `listid` parameter specifies in which list to place this new molecule. If `listid` is 0, a new list will be created to contain the created molecule.

17.5.112 MoleculeNewSubset

```
std::vector<unsigned int> MoleculeNewSubset(unsigned int listid,
                                            unsigned int id, bool pdb)
std::vector<unsigned int> MoleculeNewSubset(unsigned int listid,
                                            unsigned int id, unsigned int scope,
                                            bool unmatched)
std::vector<unsigned int> MoleculeNewSubset(unsigned int listid,
                                            unsigned int id,
                                            const std::vector<std::vector<unsigned int> >
↪ &parts)
```

This function creates a series of new molecules from the given molecule based on its internal components determined by OEChem using the OEDetermineComponents function.

17.5.113 MoleculeNewSubsetScoped

```
std::vector<unsigned int> MoleculeNewSubsetScoped(unsigned int listid,
                                                  unsigned int scope, bool pdb)
std::vector<unsigned int> MoleculeNewSubsetScoped(unsigned int listid,
                                                  unsigned int mscope,
                                                  unsigned int scope,
                                                  bool unmatched)
```

This function creates a series of new molecules from the given scope based on its internal components determined by OEChem using the OEDetermineComponents function.

17.5.114 MoleculeResidueNameSetScoped

```
void MoleculeResidueNameSetScoped(const std::string &name,
                                  unsigned int scope = BestScope)
```

Renames all the residues in the specified scope.

17.5.115 MoleculeResidueSet

```
int MoleculeResidueSet(unsigned int id, int residueNum,
                       bool incrementPerRes=false)
```

Sets the residue number of every residue in the specified molecule to the specified value (*residueNum*). The *incrementPerRes* parameter is currently reserved for future expansion.

17.5.116 MoleculeSetProperty

```
void MoleculeSetProperty(const std::vector<OEPropDB::OEKey> &keys,
                         unsigned char action, std::string prop, bool state)
```

Reserved for internal use.

17.5.117 MoleculeSizeCutoffGet

```
unsigned int MoleculeSizeCutoffGet()
```

Returns the atom number cutoff at which point a molecule is considered to be a “large” molecule. The default is 255.

17.5.118 MoleculeSizeCutoffSet

```
void MoleculeSizeCutoffSet(unsigned int size)
```

Sets the atom number cutoff at which point a molecule is considered to be a “large” molecule. The default is 255.

17.5.119 MoleculeUpdate

```
bool MoleculeUpdate(OEChem::OEMCMolBase &)
```

Updates the original molecule stored in VIDA with any changes that may have been made to the specified Python molecule.

17.5.120 NameGet

```
std::string NameGet(unsigned int)
std::string NameGet(const OEPropDB::OEKey &)
```

Returns the name of the object specified by either its ID or key.

17.5.121 NameSet

```
void NameSet(unsigned int, const std::string &)
void NameSet(const OEPropDB::OEKey &, const std::string &)
```

Sets the name of the object specified by either its ID or key.

17.5.122 OEFuseKeyGroups

```
OEKeyGroup OEFuseKeyGroups(const std::vector<OEKeyGroup> &k1,
                           unsigned char type)
```

Fuses the specified keys groups into a single key group.

17.5.123 OEKeyIterToVector

```
std::vector<OEPropDB::OEKey>
  OEKeyIterToVector(OESystem::OEIterBase<const OEPropDB::OEKey> *iterbase)
```

Converts a key iterator into a list of keys.

17.5.124 PropertyTypeGet

```
unsigned int PropertyTypeGet(const std::string &type)
```

Returns the integer property type for the string *type*. See *KeyTypeGet* for a list of types.

17.5.125 SDDataGet

```
std::string SDDataGet(unsigned int id, const std::string &tag)
std::string SDDataGet(const OEPropDB::OEKey &key, const std::string &tag)
```

Returns the SD data associated with the specified tag on the specified object.

17.5.126 SDDataHas

```
bool SDDataHas(unsigned int id, const std::string &tag)
bool SDDataHas(const OEPropDB::OEKey &key, const std::string &tag)
```

Returns whether or not the specified object contains SD data with the specified tag.

17.5.127 SDDataSet

```
void SDDataSet(unsigned int id, const std::string &tag, const std::string &data)
void SDDataSet(const OEPropDB::OEKey &key, const std::string &tag,
               const std::string &data)
```

Sets the value of the SD data associated with the specified tag on the specified object.

17.5.128 SurfaceAdd

```
unsigned int SurfaceAdd(OESpicoli::OESurface &surf, unsigned int listID = 0)
```

Adds a new Python surface object to VIDA. Returns the ID assigned to the newly added surface or zero if the add failed for any reason. If the optional listID argument is provided, then the surface is added to the specified list. Otherwise, a new list is created and the surface is added to it.

17.5.129 SurfaceBestFloodScoped

```
int SurfaceBestFloodScoped(unsigned int scope = BestScope)
```

Return the largest flood value for all surfaces in the given scope. A flood value defines the surface being scribed.

17.5.130 SurfaceCheckIn

```
bool SurfaceCheckIn(OESpicoli::OESurface &surf,
                    unsigned int checkInType = OECheckInType_UnknownChange)
```

Checks the specified Python accessible surface back into the main VIDA application which will synchronize any changes made to the surface in Python with the surface in VIDA. The surface in question must have been checked out of VIDA prior to being checked back in.

The *checkInType* parameter tells VIDA what type of changes were made to the surface in order to optimize the check in process. There are multiple allowable values for this parameter which can be OR'd together if necessary:

- OECheckInType_ConformerChange
- OECheckInType_CoordinateChange
- OECheckInType_DataChange
- OECheckInType_GraphChange
- OECheckInType_NoChange
- OECheckInType_RenderChange
- OECheckInType_AnnotationChange
- OECheckInType_UnknownChange

17.5.131 SurfaceCheckOut

```
bool SurfaceCheckOut(OESpicoli::OESurface &surf, unsigned int id)
```

Checks out a copy of the surface specified by *id* into the specified Python surface object (*surf*). Returns whether or not the check out process succeeded.

The checked out surface can be modified in Python, but those changes will not be applied to the original surface in VIDA until the modified surface is checked back in using the *SurfaceCheckIn* command.

17.5.132 SurfaceCreate

```
unsigned int SurfaceCreate(const std::string &, const OEPropDB::OEKey &key,
                           bool property=false)
```

Create a surface of type *type* based on the molecule represented by *key*.

If *property* is true, then make the surface a property surface that is bound to the molecule.

17.5.133 SurfaceCreateScoped

```
bool SurfaceCreateScoped(const std::string &, bool single_surface=false,
                         unsigned int scope = BestScope)
```

Create a surface of type *type* for each molecule in *scope*.

If *property* is true, then make the surface a property surface that is bound to the molecule.

17.5.134 SurfaceCropDistance

```
void SurfaceCropDistance(unsigned int id, float distance)
```

Retains the section of the surface specified by *oerid* to a distance *dist* from the selected set of atoms.

17.5.135 SurfaceCropDistanceFrom

```
void SurfaceCropDistanceFrom(const OEPropDB::OEKey &s, const OEPropDB::OEKey &,
                             float distance)
```

Retains the section of the surface specified by *s* based on its distance to another object. The reference object can be a molecule or another surface. The distance cutoff is measured in Angstroms.

17.5.136 SurfaceCropScribedScoped

```
void SurfaceCropScribedScoped(unsigned int scope = BestScope)
```

Retains the section of all surfaces in *scope* to a distance *dist* from the selected set of atoms.

17.5.137 SurfaceCropUnscribedScoped

```
void SurfaceCropUnscribedScoped(unsigned int scope = BestScope)
```

Remove the scribed sections of all scribed surfaces in *scope*.

17.5.138 SurfaceDelete

```
void SurfaceDelete(unsigned int id)
```

Delete the surface specified by *id*.

17.5.139 SurfaceGenerateBox

```
unsigned int SurfaceGenerateBox(float x, float y, float z, float width,
                                float height, float depth)
```

Creates and returns the id of a surface representing a box as determined by the parameters *x*, *y*, *z*, *width*, *height*, and *depth*.

17.5.140 SurfaceGenerateSphere

```
unsigned int SurfaceGenerateSphere(float x, float y, float z, float radius,
                                   int level=10)
```

Creates and returns the id of a surface representing a sphere as determined by the parameters *x*, *y*, *z*, and *radius*. The *level* parameter specifies how many iterations should be used in determining the quality of the surface.

17.5.141 SurfaceGenerateSpline

```
unsigned int SurfaceGenerateSpline(const std::vector<float> &coords,
                                   int splineResolution, int seq_res,
                                   int cross_res, float width, float thick,
                                   int splineType, int style)
```

Creates and returns the id of a surface representing a spline as determined by the parameters *coords*, *splineResolution*, *seq_res*, *cross_res*, *width*, *thick*, *splineType*, and *style*.

17.5.142 SurfacePickTriangle

```
void SurfacePickTriangle(unsigned int oerid, const OEPropDB::OEKey &trikey,
                         bool addtriangle=false)
```

Pick a surface triangle specified by *trikey*. Ordinarily, this function does not need to be called.

17.5.143 SurfaceProbeRadiusGet

```
float SurfaceProbeRadiusGet()
```

Get the probe radius used when creating surfaces.

17.5.144 SurfaceProbeRadiusSet

```
float SurfaceProbeRadiusSet(float radius)
```

Set the probe radius to *radius* that is used when creating surfaces.

17.5.145 SurfaceResolutionGet

```
float SurfaceResolutionGet()
```

Get the resolution used when creating surfaces.

17.5.146 SurfaceResolutionSet

```
float SurfaceResolutionSet(float res)
```

Set the resolution used when creating surfaces. The resolution is set to *resolution*.

17.5.147 SurfaceRestoreScoped

```
void SurfaceRestoreScoped(unsigned int scope = BestScope)
```

Restore all surfaces in *scope* to their original states.

17.5.148 SurfaceScribeScoped

```
void SurfaceScribeScoped(int flood,
                         unsigned int scope = BestScope)
```

Scribe all scribed surfaces in scope to include all portions of the surface that fall within *flood*. Ordinarily, this function does not need to be called.

17.5.149 SurfaceSetPotentialFromGrid

```
void SurfaceSetPotentialFromGrid(unsigned int id, unsigned int gridid)
```

Sets the surface potential for the specified surface using the values in the specified grid.

17.5.150 SurfaceSetPotentialFromGridScoped

```
void SurfaceSetPotentialFromGridScoped(unsigned int gridid,
                                       unsigned int scope = BestScope)
```

Sets the surface potential for all the surfaces in the specified scope using the values in the specified grid.

17.5.151 SurfaceVolume

```
float SurfaceVolume(unsigned int id)
```

Returns the enclosed volume of surface *id*.

17.5.152 SymmetryNumOperators

```
unsigned int SymmetryNumOperators(unsigned int id)
```

This function returns the number of symmetry operators for the specified id.

17.5.153 SymmetryOperatorEnabledGet

```
bool SymmetryOperatorEnabledGet(unsigned int id, unsigned int operatorNumber)
```

17.5.154 SymmetryOperatorEnabledSet

```
bool SymmetryOperatorEnabledSet(unsigned int id, unsigned int operatorNumber,
                                bool enable)
```

17.5.155 SymmetryRadiusGet

```
float SymmetryRadiusGet()
```

17.5.156 SymmetryRadiusSet

```
float SymmetryRadiusSet(float r)
```

17.5.157 SymmetryRealize

```
unsigned int SymmetryRealize(unsigned int id, bool make_new_mol,
                             std::vector<float> center=std::vector<float>(),
                             float radius=0.0f)
```

17.5.158 XRayAutoMapCalculationPrefsSet

```
void XRayAutoMapCalculationPrefsSet()
void XRayAutoMapCalculationPrefsSet(const std::string &map1,
                                                const std::string &map2="")
```

Set the map types for maps which should be calculated automatically after reading a reflection file.

17.5.159 XRayCalculateMap

```
unsigned int XRayCalculateMap(unsigned int map_id,
                              unsigned int refln_id,
                              std::string map_type)
unsigned int XRayCalculateMap(unsigned int map_id,
                              unsigned int refln_id,
                              unsigned int map_type)
```

This set of overloaded functions calculates a map for the set of reflection data specified by 'refln_id'. If 'map_id' is non-zero, then the map replaces the grid with that ID. Otherwise, an entirely new map is created. The string 'map_type' parameter may be one of: Fo, Fc, 2Fo-Fc, Fo-Fc, 3Fo-2Fc, 5Fo-3Fc, Fo*Fo, or Fo*FOM, while the integer version may be one of: 0 Fo, 1 Fc, 2 TwoFoFc, 3 FoFc, 4 ThreeFoTwoFc, 5 FiveFoThreeFc, 8 FoSquared, 9 FoFom.

17.5.160 XRayCalculatePhases

```
bool XRayCalculatePhases(unsigned int refln_id,
                         unsigned int model_id)
```

This is experimental, and should not be called by the user.

17.5.161 XRayGetCell

```
std::vector<float> XRayGetCell(unsigned int id)
```

Returns the cell constants for the given object in a list, in the order a, b, c, Alpha, Beta, Gamma.

17.5.162 XRayGetSpaceGroup

```
std::string XRayGetSpaceGroup(unsigned int id)
```

Returns the spacegroup name for the given object.

17.5.163 XRayMTZColumnNamesCurrentDefaultsGet

```
std::vector<std::string> XRayMTZColumnNamesCurrentDefaultsGet()
```

17.5.164 XRayMTZColumnNamesGet

```
std::vector<std::string> XRayMTZColumnNamesGet()
bool XRayMTZColumnNamesGet(std::vector<std::string> &names)
```

Fills up the input list/vector with the MTZ file column names as specified by *XRayMTZColumnNamesSet*. Unless *XRayMTZColumnNamesSet* was called with the *permanent* flag, this function will only work once per call to *XRayMTZ-ColumnNamesSet*.

This function returns True if it successfully filled out the list/vector, or False otherwise.

17.5.165 XRayMTZColumnNamesSet

```
void XRayMTZColumnNamesSet(const std::vector<std::string> &names,
                           bool permanent=false)
```

Stores a list of column names for mtz files. The next time an MTZ file is read, instead of prompting the user, the column names are taken from the list/vector passed to this function. The column names are in the order Fo, Phi, Fc, PhiC, Sigma, FOM, RFree. If the *permanent* flag is set to True, then these values are used instead of prompting until the program exits. If *permanent* is false, then the values are only used once.

In a script, there's no reason to call this command instead of pre-specifying the prompt results with *PromptResponseVectMulti*, as shows up in the journal file. However, if creating a function which will read an MTZ file directly then this command is needed since the *PromptResponse* commands don't work in interactive commands.

17.5.166 XRayMTZColumnNamesStandardDefaultsGet

```
std::vector<std::string> XRayMTZColumnNamesStandardDefaultsGet()
```

17.5.167 XRaySetCrystalParams

```
bool XRaySetCrystalParams(unsigned int id,
                          unsigned int from_object_id)
bool XRaySetCrystalParams(unsigned int id,
                          std::vector<float> cell,
                          unsigned int sg)
```

(continues on next page)

(continued from previous page)

```
bool XRaySetCrystalParams(unsigned int id,
                          std::vector<float> cell,
                          const std::string &sgname)
bool XRaySetCrystalParams(unsigned int id, float a, float b,
                          float c, float alpha, float beta,
                          float gamma, unsigned int sg)
bool XRaySetCrystalParams(unsigned int id, float a, float b,
                          float c, float alpha, float beta,
                          float gamma, const std::string &sgname)
```

Set the crystal parameters for the given reflection. The parameters may be specified in multiple ways, which are obvious from the names of the arguments.

17.5.168 XRayValidMaptypes

```
std::string XRayValidMaptypes(unsigned int refln_id)
```

Not all map types are available for every reflection data set, since not all files have all required data for every possible map calculation. For example, a reflection data set without Fc data will not be able to calculate any map types which require Fc. This function returns (as a string), the valid map types for the given reflection data set. The string is a concatenation of all available map types, with each type separated by the '|' character.

17.6 Molecule Builder Functions

The functions detailed in this section provide the ability to create new molecules from basic inputs and to modify existing ones.

17.6.1 AtomAddHydrogens

```
void AtomAddHydrogens(const OEPropDB::OEKey &k, bool polaronly=false)
```

Adds explicit hydrogens to the specified atom. If *polaronly* is set to True, this will only add polar hydrogens to the specified atom.

17.6.2 AtomAddHydrogensScoped

```
void AtomAddHydrogensScoped(bool polaronly=false,
                            unsigned int scope = SelectedScope)
```

Scoped version of *AtomAddHydrogens*.

17.6.3 AtomAtomicNumDefaultGet

```
unsigned int AtomAtomicNumDefaultGet()
```

Returns the default value of the atomic number to be used when setting the atomic number of an atom via a mouse action.

17.6.4 AtomAtomicNumDefaultSet

```
void AtomAtomicNumDefaultSet(unsigned int)
```

Sets the default value of the atomic number to be used when setting the atomic number of an atom via a mouse action.

17.6.5 AtomAtomicNumGet

```
unsigned int AtomAtomicNumGet(const OEPropDB::OEKey &k)
```

Returns the atomic number of the specified atom.

17.6.6 AtomAtomicNumSet

```
void AtomAtomicNumSet(const OEPropDB::OEKey &k, unsigned int an)
```

Sets the atomic number of the specific atom.

17.6.7 AtomAtomicNumSetScoped

```
void AtomAtomicNumSetScoped(unsigned int an,
                            unsigned int scope = SelectedScope)
```

Scoped version of *AtomAtomicNumSet*.

17.6.8 AtomAttach

```
void AtomAttach(const OEPropDB::OEKey &k, const std::string &oeb)
void AtomAttach(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2)
void AtomAttach(const OEPropDB::OEKey &k, const std::string &smiles, int attach)
```

Attaches the specified molecule fragment to the specified atom. There are three different implementations of this function which allow for different specifications of the fragment. The atom being attached to, is always the first parameter. The attachment is performed by the creation of a single bond.

The first implementation takes a hexadecimal encoding of a molecule specified in OEB format. This implementation is primarily intended for internal use.

The second implementation takes an OEKey parameter specifying an atom on an already loaded fragment to which the attachment will occur.

The third implementation takes a SMILES string representing the fragment to be attached as well as an index specifying which atom in that fragment will be the attachment point.

17.6.9 AtomDelete

```
void AtomDelete(const OEPropDB::OEKey &k)
```

Deletes the specified atom as well as any attached hydrogens.

17.6.10 AtomDeleteHydrogens

```
void AtomDeleteHydrogens(const OEPropDB::OEKey &k)
```

Deletes all of the hydrogens attached to the specified atom.

17.6.11 AtomDeleteHydrogensScoped

```
void AtomDeleteHydrogensScoped(bool nonpolar=false,
                               unsigned int scope = SelectedScope)
```

Scoped version of *AtomDeleteHydrogens*.

17.6.12 AtomDeleteScoped

```
void AtomDeleteScoped(unsigned int scope = SelectedScope)
```

Scoped version of *AtomDelete*.

17.6.13 AtomFormalChargeDefaultGet

```
int AtomFormalChargeDefaultGet()
```

Returns the default value of the formal to be used when setting the formal of an atom via a mouse action.

17.6.14 AtomFormalChargeDefaultSet

```
void AtomFormalChargeDefaultSet(int)
```

Sets the default value of the formal to be used when setting the formal of an atom via a mouse action.

17.6.15 AtomFormalChargeGet

```
int AtomFormalChargeGet(const OEPropDB::OEKey &k)
```

Returns the formal charge for the specified atom.

17.6.16 AtomFormalChargeModify

```
void AtomFormalChargeModify(const OEPropDB::OEKey &k, int delta)
```

Modifies the formal charge on the specified atom by the specified delta value. The delta value can be either positive or negative.

17.6.17 AtomFormalChargeModifyScoped

```
void AtomFormalChargeModifyScoped(int delta,
                                  unsigned int scope = SelectedScope)
```

Scoped version of *AtomFormalChargeModify*.

17.6.18 AtomFormalChargeSet

```
void AtomFormalChargeSet(const OEPropDB::OEKey &k, int chg)
```

Sets the formal charge on the specified atom.

17.6.19 AtomFormalChargeSetScoped

```
void AtomFormalChargeSetScoped(int chg,
                               unsigned int scope = SelectedScope)
```

Scoped version of *AtomFormalChargeSet*.

17.6.20 AtomFuse

```
void AtomFuse(const OEPropDB::OEKey &k, const std::string &oeb)
void AtomFuse(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2)
void AtomFuse(const OEPropDB::OEKey &k, const std::string &smiles, int attach)
```

Fuses the specified molecule fragment to the specified atom. There are three different implementations of this function which allow for different specifications of the fragment. The atom being fused to is always the first parameter.

The first implementation takes a hexadecimal encoding of a molecule specified in OEB format. This implementation is primarily intended for internal use.

The second implementation takes an OEKey parameter specifying an atom on an already loaded fragment where the fusion will occur.

The third implementation takes a SMILES string representing the fragment to be fused as well as an index specifying which atom in that fragment will be the site of the fusion.

17.6.21 AtomHybridizationGet

```
unsigned int AtomHybridizationGet(const OEPropDB::OEKey &k)
```

Returns the hybridization state of the specified atom.

17.6.22 AtomHybridizationSet

```
void AtomHybridizationSet(const OEPropDB::OEKey &k, unsigned int hyb)
```

Sets the hybridization state of the specified atom.

17.6.23 AtomHybridizationSetScoped

```
void AtomHybridizationSetScoped(unsigned int hyb,
                                unsigned int scope = SelectedScope)
```

Scoped version of *AtomHybridizationSet*.

17.6.24 AtomIsotopeGet

```
unsigned int AtomIsotopeGet(const OEPropDB::OEKey &k)
```

Returns the isotope of the specified atom.

17.6.25 AtomIsotopeSet

```
void AtomIsotopeSet(const OEPropDB::OEKey &k, unsigned int iso)
```

Sets the isotope for the specified atom.

17.6.26 AtomIsotopeSetScoped

```
void AtomIsotopeSetScoped(unsigned int iso,
                          unsigned int scope = SelectedScope)
```

Scoped version of *AtomIsotopeSet*.

17.6.27 AtomSprout

```
void AtomSprout(const OEPropDB::OEKey &k, unsigned int elem=6,
                unsigned int order=1)
```

Sprouts a new atom off of the specified atom. The atomic number of the new atom is specified by *elem* and the order of the bond to the new atom is specified by the *order* parameter.

17.6.28 AtomSproutScoped

```
void AtomSproutScoped(unsigned int elem=6, unsigned int order=1,
                      unsigned int scope = SelectedScope)
```

Scoped version of *AtomSprout*.

17.6.29 AtomStereoDefaultGet

```
std::string AtomStereoDefaultGet()
```

Returns the default value for the stereochemistry setting to be used when setting the stereochemistry via a mouse action. Allowed values include:

- "R"
- "S"
- "Invert"

17.6.30 AtomStereoDefaultSet

```
void AtomStereoDefaultSet(const std::string &s)
```

Sets the default value for the stereochemistry setting to be used when setting the stereochemistry via a mouse action. Allowed values include:

- "R"
- "S"
- "Invert"

17.6.31 AtomStereoSet

```
void AtomStereoSet(const OEPropDB::OEKey &k, const std::string &s)
```

Sets the CIP stereochemistry of the specified atom. Allowed values are "R" and "S".

17.6.32 AtomStereoSetScoped

```
void AtomStereoSetScoped(const std::string &s,
                         unsigned int scope = SelectedScope)
```

Scoped version of *AtomStereoSet*.

17.6.33 AtomStereoToggle

```
void AtomStereoToggle(const OEPropDB::OEKey &k)
```

Toggles the stereochemistry of the specified atom.

17.6.34 AtomStereoToggleScoped

```
void AtomStereoToggleScoped(unsigned int scope = SelectedScope)
```

Scoped version of: *AtomStereoToggle*.

17.6.35 BondAngleGet

```
double BondAngleGet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2)
double BondAngleGet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                    const OEPropDB::OEKey &k3)
```

Returns the angle between the two specified bonds or the three specified atoms.

17.6.36 BondAngleModify

```
void BondAngleModify(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                     double angle, unsigned int rmode)
void BondAngleModify(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                     const OEPropDB::OEKey &k3, double ang, unsigned int rmode)
```

Modifies the angle between the two specified bonds or the three specified atoms. The amount by which the angle is adjusted is specified by the *angle* parameter. The portion of the molecule which is actually moved in this process is specified by the *rmode* parameter:

- 0 - the smaller portion of the molecule will be moved
- 1 - the larger portion of the molecule will be moved

17.6.37 BondAngleSet

```
void BondAngleSet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                  double angle, unsigned int rmode)
void BondAngleSet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                  const OEPropDB::OEKey &k3, double ang, unsigned int rmode)
```

Sets the angle between the two specified bonds or the three specified atoms. The desired angle is specified by the *angle* parameter. The portion of the molecule which is actually moved in this process is specified by the *rmode* parameter:

- 0 - the smaller portion of the molecule will be moved
- 1 - the larger portion of the molecule will be moved

17.6.38 BondCreate

```
void BondCreate(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                unsigned int order=1, double length=0.0)
```

Creates a new bond between the two specified atoms with an order and length specified by the two eponymous parameters.

17.6.39 BondDelete

```
void BondDelete(const OEPropDB::OEKey &k)
```

Deletes the specified bond.

17.6.40 BondDeleteScoped

```
void BondDeleteScoped(unsigned int scope = SelectedScope)
```

Scoped version of *BondDelete*.

17.6.41 BondFuse

```
void BondFuse(const OEPropDB::OEKey &b, const std::string &oeb)
void BondFuse(const OEPropDB::OEKey &b1, const OEPropDB::OEKey &b2)
void BondFuse(const OEPropDB::OEKey &b, const OEPropDB::OEKey &e,
              const std::string &oeb)
void BondFuse(const OEPropDB::OEKey &b, const std::string &smiles, int battach,
              int eattach)
void BondFuse(const OEPropDB::OEKey &b1, const OEPropDB::OEKey &e1,
              const OEPropDB::OEKey &b2, const OEPropDB::OEKey &e2)
void BondFuse(const OEPropDB::OEKey &beg, const OEPropDB::OEKey &end,
              const std::string &smiles, int battach, int eattach)
```

Fuses two bonds together. There are multiple implementations of this function which allow for different mechanisms of specifying the source bond as well as the target bond (and associated fragment). The source bond can be specified using a distinct bond key or by specifying the keys of the beginning and end atoms of that bond. The target bond can also be specified using a distinct bond key or by two keys representing the beginning and end atoms of the bond. However, the target bond can also be set as part of a specified fragment.

One implementation allows for specification of a hexadecimal encoded OEB string representing the molecule fragment and selected target bond. This method is primarily intended for internal use.

The other mechanism allows for specification of the fragment as a SMILES string associated with an index for the beginning and end atoms defining the bond.

17.6.42 BondLengthGet

```
double BondLengthGet(const OEPropDB::OEKey &k)
double BondLengthGet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2)
```

Returns the length of the specified bond. The bond can be specified using either a bond key or two atom keys corresponding to the endpoints of the bond.

17.6.43 BondLengthModify

```
void BondLengthModify(const OEPropDB::OEKey &k, double len, unsigned int rmode)
void BondLengthModify(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                      double length, unsigned int rmode)
```

Modifies the length of the specified bond by the amount specified in the *len* parameter. The bond can be specified using either a bond key or two atom keys corresponding to the endpoints of the bond. The portion of the molecule which is actually moved is determined by the *rmode* parameter. Available values include:

- 0 - the smaller portion of the molecule is moved
- 1 - the larger portion of the molecule is moved

17.6.44 BondLengthSet

```
void BondLengthSet(const OEPropDB::OEKey &k, double len, unsigned int rmode)
void BondLengthSet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                   double length, unsigned int rmode)
```

Sets the length of the specified bond to the value specified in the *len* parameter. The bond can be specified using either a bond key or two atom keys corresponding to the endpoints of the bond. The portion of the molecule which is actually moved is determined by the *rmode* parameter. Available values include:

- 0 - the smaller portion of the molecule is moved
- 1 - the larger portion of the molecule is moved

17.6.45 BondLengthSetScoped

```
void BondLengthSetScoped(double length,
                         unsigned int rmode=OEForge::RotateMode::Smaller,
                         unsigned int scope = SelectedScope)
```

Scoped version of *BondLengthSet*.

17.6.46 BondOrderDefaultGet

```
unsigned int BondOrderDefaultGet()
```

Returns the default value of the bond order used when setting the order of a bond through a mouse action.

17.6.47 BondOrderDefaultSet

```
void BondOrderDefaultSet(unsigned int)
```

Sets the default value of the bond order used when setting the order of a bond through a mouse action.

17.6.48 BondOrderGet

```
unsigned int BondOrderGet(const OEPropDB::OEKey &k)
unsigned int BondOrderGet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2)
```

Returns the order of the specified bond. The bond can be specified as a distinct bond key or two keys corresponding to the endpoint atoms.

17.6.49 BondOrderSet

```
void BondOrderSet(const OEPropDB::OEKey &k, unsigned int order)
void BondOrderSet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                  unsigned int order)
```

Sets the order of the specified bond. The bond can be specified as a distinct bond key or two keys corresponding to the endpoint atoms.

17.6.50 BondOrderSetScoped

```
void BondOrderSetScoped(unsigned int order, unsigned int scope)
```

Scoped version of *BondOrderSet*.

17.6.51 BondStereoDefaultGet

```
std::string BondStereoDefaultGet()
```

Returns the default value for bond stereochemistry setting to be used when setting the stereochemistry via a mouse action. Allowed values include:

- "E"
- "Z"
- "Invert"

17.6.52 BondStereoDefaultSet

```
void BondStereoDefaultSet(const std::string &)
```

Sets the default value for bond stereochemistry setting to be used when setting the stereochemistry via a mouse action. Allowed values include:

- "E"
- "Z"
- "Invert"

17.6.53 BondStereoSet

```
void BondStereoSet(const OEPropDB::OEKey &k, const std::string &s)
void BondStereoSet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                   const std::string &s)
```

Sets the CIP stereochemistry for the specified bond. The bond can be specified as a distinct bond key or by two atom keys corresponding to the endpoint atoms. Allowed values for stereochemistry include:

- "E"
- "Z"

17.6.54 BondStereoSetScoped

```
void BondStereoSetScoped(const std::string &s,
                         unsigned int scope = SelectedScope)
```

Scoped version of *BondStereoSet*.

17.6.55 BondStereoToggle

```
void BondStereoToggle(const OEPropDB::OEKey &k)
void BondStereoToggle(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2)
```

Toggles the stereochemistry for the specified bond. The bond can be specified as a distinct bond key or by two atom keys corresponding to the endpoint atoms.

17.6.56 BondStereoToggleScoped

```
void BondStereoToggleScoped(unsigned int scope = SelectedScope)
```

Scoped version of *BondStereoToggle*.

17.6.57 BondTorsionGet

```
double BondTorsionGet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                      const OEPropDB::OEKey &k3)
double BondTorsionGet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                      const OEPropDB::OEKey &k3, const OEPropDB::OEKey &k4)
```

Returns the torsion (dihedral) angle specified by either three bonds or four atoms. The order of specification is important in the determination of the angle.

17.6.58 BondTorsionModify

```
void BondTorsionModify(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                       const OEPropDB::OEKey &k3, double t, unsigned int rmode)
void BondTorsionModify(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                       const OEPropDB::OEKey &k3, const OEPropDB::OEKey &k4,
                       double torsion, unsigned int rmode)
```

Modifies the torsion (dihedral) angle specified by either three bonds or four atoms. The amount the angle is modified by is specified by the *torsion* parameter. The portion of the molecule which is moved by this function is specified by the *rmode* parameter. Available values include:

- 0 - the smaller portion of the molecule moves
- 1 - the larger portion of the molecule moves

17.6.59 BondTorsionSet

```
void BondTorsionSet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                    const OEPropDB::OEKey &k3, double t, unsigned int rmode)
void BondTorsionSet(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                    const OEPropDB::OEKey &k3, const OEPropDB::OEKey &k4,
                    double torsion, unsigned int rmode)
```

Sets the torsion (dihedral) angle specified by either three bonds or four atoms. The desired angle is specified by the *torsion* parameter. The portion of the molecule which is moved by this function is specified by the *rmode* parameter. Available values include:

- 0 - the smaller portion of the molecule moves
- 1 - the larger portion of the molecule moves

17.6.60 BuilderActiveGet

```
bool BuilderActiveGet()
```

Returns whether or not VIDA is in molecule editing mode.

17.6.61 BuilderActiveSet

```
void BuilderActiveSet(bool)
```

Sets whether or not VIDA is in molecule editing mode.

17.6.62 BuilderAlternateConfModeGet

```
bool BuilderAlternateConfModeGet()
```

Returns whether or not VIDA is in ring alternate conformation exploration mode.

17.6.63 BuilderAlternateConfModeSet

```
void BuilderAlternateConfModeSet(bool)
```

Sets whether or not VIDA is in ring alternate conformation exploration mode.

17.6.64 BuilderAlternateConfSet

```
void BuilderAlternateConfSet(unsigned int)
```

Applies the specified alternate ring conformation to the molecule currently being edited. This is intended for internal use only.

17.6.65 BuilderCancel

```
void BuilderCancel()
```

Cancels all modifications made to the molecule currently being edited and exits editing mode.

17.6.66 BuilderEdit

```
void BuilderEdit(const OEPropDB::OEKey &, bool force3D)
```

Puts VIDA into editing mode for the specified molecule. If the molecule is a 2D molecule, setting the *force3D* parameter to True will ensure that 3D coordinates will be generated, even if the user cancels out of editing mode.

17.6.67 BuilderFinish

```
void BuilderFinish()
```

Commits all modifications made to the molecule currently being edited and exits editing mode.

17.6.68 BuilderFragmentGet

```
std::string BuilderFragmentGet()
```

Returns a hexadecimal encoded OEB string representation of the current fragment being used in the builder component.

17.6.69 BuilderFragmentSet

```
void BuilderFragmentSet(const std::string &oeb)
void BuilderFragmentSet(const std::string &smiles, int)
```

Sets the default fragment to be used in mouse actions in the builder. The first implementation takes a hexadecimal encoded OEB string representation of the molecule fragment (this is primarily intended for internal use). The second implementation takes a SMILES string with an additional index parameter which specifies the index of the atom to be used as the attachment point.

17.6.70 BuilderMinimize

```
void BuilderMinimize()
```

Minimizes the molecule currently being edited using the MMFF forcefield.

17.6.71 BuilderMinimizeScoped

```
void BuilderMinimizeScoped(unsigned int scope = SelectedScope)
```

Minimizes a portion of the molecule currently being edited as determined by the atoms in the specified scope. Valid values for the scope parameter are:

- MarkedScope
- SelectedScope
- ScratchScope

17.6.72 BuilderPropertiesGet

```
std::vector<string> BuilderPropertiesGet()
```

Returns a list of the properties displayed in the Property table of the Builder window.

17.6.73 BuilderPropertyAdd

```
void BuilderPropertyAdd(const std::string &name, const std::string &command, unsigned␣
↪int flags = CalculateForSmallMols)
```

Adds a user-defined property calculator to the Properties table of the Builder window. The *name* parameter is the name of the property to be calculated. The *command* parameter is a string containing a Python command to be executed to calculate the property. The return value of the command is displayed in the property table. The *flags* parameter specifies what types of molecules the property will be calculated for, and whether a change to a molecule's coordinates requires the property to be recalculated. Valid values for this parameter are:

- UpdateOnCoordChange
- CalculateForSmallMols
- CalculateForProteins

These values can be OR'ed together to be used in combinations. The property is always recalculated when the chemical graph of the active molecule changes.

To add one of VIDA's default properties (*e.g.*, 'LogP') to the property table, use its name and leave the *command* string empty.

17.6.74 BuilderPropertyRemove

```
void BuilderPropertyRemove(const std::string &propertyToRemove)
```

Removes a property from the Property table of the Builder window.

17.6.75 BuilderPropertySetValue

```
void BuilderPropertySetValue(const std::string &propertyName, const std::string &value)
```

Directly sets a value in the Property table of the Builder window. The value will be overwritten when the property is updated.

17.6.76 BuilderTorsionActivate

```
void BuilderTorsionActivate(const OEPropDB::OEKey &key)
```

Activates an existing torsion monitor, specified by key, so that it can be rotated interactively while in Editing mode. Activating a torsion will temporarily display two torsion measurements, one for each end of the bond. When one of these measurements is selected, the corresponding end of the molecule will be moved when the mouse wheel or up/down arrows are used. If this command is used outside of Editing mode, the double measurements will be displayed but will not be activated.

17.6.77 BuilderTorsionsDeactivate

```
void BuilderTorsionsDeactivate()
```

Returns all active torsion monitors to their inactive state.

17.6.78 BuilderUseMMFF94sSet

```
void BuilderUseMMFF94sSet(bool useMMFF94s)
```

Causes the builder geometry optimizations to use the MMFF94s force field, which provides more nearly planar geometries for delocalized trigonal nitrogens than the MMFF94 force field. Calling this function with *useMMFF94s* = False will cause the builder to revert to using MMFF94.

17.6.79 BuilderUseMMFF94sGet

```
bool BuilderUseMMFF94sGet()
```

Returns True if the builder is currently set to use the MMFF94s force field, or False if the builder is using the MMFF94 force field.

17.6.80 MoleculeAddExplicitHydrogens

```
void MoleculeAddExplicitHydrogens(const OEPropDB::OEKey &k,
                                  bool polarOnly=false)
```

Adds explicit hydrogens to the specified molecule. If the *polarOnly* parameter is specified, only polar hydrogens will be added.

17.6.81 MoleculeAddHydrogens

```
void MoleculeAddHydrogens(const OEPropDB::OEKey &k, bool polarOnly=false)
```

This function adds hydrogens to the molecule associated with the specified key. If the *polarOnly* parameter is set, only polar hydrogens will be added.

17.6.82 MoleculeAddHydrogensScoped

```
void MoleculeAddHydrogensScoped(bool polarOnly=false,
                                unsigned int scope = BestScope)
```

This function adds hydrogens to the molecules int the specified scope. If the *polarOnly* parameter is set, only polar hydrogens will be added.

17.6.83 MoleculeDeleteHydrogens

```
void MoleculeDeleteHydrogens(const OEPropDB::OEKey &k, bool nonpolar=false)
```

Deletes hydrogens off of the specified molecule. If the *nonpolar* parameter is True, it will only delete non-polar hydrogens, leaving the molecule with only polar hydrogens still attached.

17.6.84 MoleculeGenerateCoords

```
void MoleculeGenerateCoords(const OEPropDB::OEKey &k,
                            unsigned int maxconfs=1,
                            bool strict=false)
```

Generates new coordinates for the specified molecule. The maximum number of conformations can be specified using the *maxconfs* parameter. Strict stereo can be turn on using the *strict* parameter.

17.6.85 MoleculeGenerateCoordsFixed

```
void MoleculeGenerateCoordsFixed(const OEPropDB::OEKey &k,
                                 const std::vector<OEPropDB::OEKey> &fixed,
                                 bool match=true, unsigned int maxconfs=1,
                                 bool strict=false)
```

Generates new coordinates for the specified molecule while holding fixed either the specified atoms (if the *match* parameter is True) or all the unspecified atoms (if the *match* parameter is False). The maximum number of conformations can be specified use the *maxconfs* parameter. Strict stereo can be turn on using the *strict* parameter.

17.6.86 MoleculeGenerateCoordsFixedScoped

```
void MoleculeGenerateCoordsFixedScoped(const OEPropDB::OEKey &k,
                                       bool match=true, unsigned int maxconfs=1,
                                       unsigned int scope = SelectedScope,
                                       bool strict)
```

Generates new coordinates for the specified molecule while holding fixed either all the atoms in the specified scope (if the *match* parameter is True) or all the atoms not in specified scope (if the *match* parameter is False). The maximum number of conformations can be specified use the *maxconfs* parameter. Strict stereo can be turn on using the *strict* parameter.

17.6.87 MoleculeNew

```
OEPropDB::OEKey MoleculeNew(const std::string &text, unsigned int listid=0)
```

Creates a new molecule from the specified text representation. Allowed text representations include SMILES as well as IUPAC and common names. A FASTA sequence can also be specified if preceded by an angle bracket character ('>').

Assuming that the molecule can be created from the specified text, it will be added to the specified list (*listid*). If the list ID is zero, a new list will be created.

Returns the key corresponding to this molecule.

17.6.88 MoleculeRotate

```
void MoleculeRotate(const OEPropDB::OEKey &key,
                    int oldx, oldy, int newx, int newy)
```

Rotates a molecule's coordinates corresponding to a mouse move in the 3D display window.

17.6.89 SketcherInputSet

```
void SketcherInputSet(const std::string &smilesString)
void SketcherInputSet(OEMol &molecule)
```

Sets the molecule in the Sketcher from the provided SMILES string or molecule. If an existing 3D molecule is already in build mode, this method will throw an exception.

17.6.90 SketcherLookupFunctionSet

```
void SketcherLookupFunctionSet(const std::string &functionString)
```

Sets a Python function to be called for compound name lookups. The lookup function should accept a string, convert that to either SMILES or a molecule, and call *SketcherInputSet* to put the resulting molecule into the Sketcher. The `functionString` argument should include only the name of the lookup function, without parentheses or arguments, *e.g.*, 'MyLookupFunction'.

17.7 Data Analysis Functions

The functions detailed in this section provide access to the data analysis and spreadsheet functionality in VIDA. An important distinction should be made in that the "Datatable" functions refer to internal data collections while the "Spreadsheet" functions refer to the actual tabular views of the associated data that actually appear in the Spreadsheet display.

17.7.1 DataAdd

```
void DataAdd(const OEPropDB::OEKey &key, const std::string &str,
             const std::string &data)
```

Adds a piece of data to the object specified by *key*. If a column in the spreadsheet does not already exist for this data type, it will be added.

17.7.2 DataGetDB

```
VFDataBase *DataGetDB()
```

Return the VDDataBase object.

17.7.3 DataGetTable

```
VFDataTable *DataGetTable(const std::string &name, bool throwError=true)
```

Return the list of all internal databases.

17.7.4 DatatableAddColumn

```
void DatatableAddColumn(std::string datatable, std::string columnName,
                        bool genericData=false)
```

Add a column to the specified data table with the specified name. If a column with the specified name already exists, this call will be ignored.

17.7.5 DatatableCommitChanges

```
void DatatableCommitChanges(const std::string &datatable)
```

Commits any outstanding changes to the internal database.

17.7.6 DatatableCurrentGet

```
std::string DatatableCurrentGet()
```

Returns the name of the current data table (e.g. 'Molecules').

17.7.7 DatatableCurrentSet

```
bool DatatableCurrentSet(const std::string &datatable)
```

Sets the current data table.

17.7.8 DatatableData

```
std::string DatatableData(const std::string &datatable, unsigned int row,
                          std::string header)
```

Returns the string representation of the data in the specified data table at the specified row and column.

17.7.9 DatatableDeleteColumn

```
void DatatableDeleteColumn(const std::string &datatable,
                           const std::string &name)
```

Delete the specified column from the specified data table.

17.7.10 DatatableEditableGet

```
bool DatatableEditableGet(const std::string &ss)
```

Returns whether or not the specified data table is editable.

17.7.11 DatatableEditableSet

```
void DatatableEditableSet(const std::string &ss, bool val)
```

Sets whether or not the specified data table is editable.

17.7.12 DatatableFilter

```
void DatatableFilter(const std::string &datatable, const std::string &filter,
                     const std::string &name, bool staticView=true,
                     bool permaFilter=false)

void DatatableFilter(const std::string &datatable, const OEDataFilter &filter,
                     const std::string &name, bool staticView=true,
                     bool permaFilter=false)
```

Creates a new data table view with the specified name from the specified data table using the specified filter. The filter is an arbitrary Python expression that is applied to every row in the source data table to determine whether or not it should be included in the filtered view.

The filter expression may assume that a local variable `ROW` is set to the current row number being evaluated. An example appears below:

```
 int(DatatableData(DatatableCurrent(), ROW, foo)) < 400

The example shows an example expression where the string representation of
the data in the current data table (which may not necessarily be the
specified datatable) at row ``ROW`` and in column ``foo`` is converted to an
integer and compared with the value 400. If this expression is ``True`` for
the row being tested, it will be included in the filtered view, otherwise it
will not be included.
```

17.7.13 DatatableFromList

```
void DatatableFromList(unsigned int listid, const std::string &name)
```

Creates a filtered data table with the specified name with row entries for each object in the specified list. If the specified name already exists, a new name will be used (e.g. *name* –> *name1*, *name2*, and so on).

17.7.14 DatatableGetColumn

```
std::vector<std::string> DatatableGetColumn(const std::string &datatable,
                                            unsigned int index)

std::vector<std::string> DatatableGetColumn(const std::string &datatable,
                                            const std::string &name)
```

Returns a list of string representations for each entry in the specified column in the specified data table.

17.7.15 DatatableGetCurrentRow

```
int DatatableGetCurrentRow(std::string datatable)
```

Returns the row index for the first active or selected row in the specified data table.

17.7.16 DatatableGetDatatables

```
std::vector<std::string> DatatableGetDatatables()
```

Returns a list of the names of all the available datatables.

17.7.17 DatatableGetImageStreamAtRow

```
bool DatatableGetImageStreamAtRow(const std::string &datatable,
                                  const std::string &filename, int row,
                                  int width=256, int height=256)
```

Write an image file to the file specified by *filename* for the datatable named *datatable* at the row specified by *row*. The size of the image is specified by *width* and *height*.

17.7.18 DatatableGetKeys

```
std::vector<OEKey> DatatableGetKeys(const std::string &datatable)
```

Returns a list of keys for the database specified by *datatable*. The keys are returned in the current datatable sort order and are consistent with the ordering from the call to DatatableGetColumn.

17.7.19 DatatableGetNumRows

```
int DatatableGetNumRows(std::string datatable)
```

Return the number of rows for the specified datatable.

17.7.20 DatatableHeaders

```
std::vector<std::string> DatatableHeaders(const std::string &datatable)
```

Return a list of all the names of all the headers for the specified datatable.

17.7.21 DatatableLingoSimSort

```
bool DatatableLingoSimSort(const std::string &spreadsheet, unsigned int row)
```

Sorts the specified datatable according to Lingo similarity to the molecule contained in the specified row.

17.7.22 DatatableMolNumberFunction

```
double DatatableMolNumberFunction(const std::string &datatable, int row,
                                  const std::string &func)
```

Calculates molecular data that returns a numeric value. *datatable* identifies the datatable containing the molecule and row is the datatable row with the molecule.

The function *func* to compute is one of:

- "mw" Molecular weight.
- "Num Atoms" Number of atoms in the molecule.
- "Num Bonds" number of bonds in the molecule.
- **"Carbon-Hetero ratio" the ratio of carbons to hetero atoms in the molecule.**
  Returns -1 if there are no carbons.
- "Energy" The molecular energy of the molecule as specified in the input file.
- "Actual Charge" The sum of the partial charges on all atoms as specified in the input file.
- "Formal Charge" The sum of the formal charges on all atoms.
- "Halide Count" The number of halogen atoms in the molecule.
- "Num Carbons" The number of carbon atoms in the molecule.
- "Num Formal Charges" The number of more atoms with a specified formal charge.
- "Num Heavy Atoms" The number of heavy atoms (non hydrogen) in the molecule.
- "Num Hetero Atoms" The number of hetero atoms in the molecule.
- "Num Hydrogens" The number of hydrogen atoms in the molecule.
- "Num Rigid Bonds" The number of rigid bonds in the molecule.
- "Nom Rotatable Bonds" The number of rotatable bonds in the molecule.
- "Num Chiral Atoms" returns the number of chiral atoms in the molecule.
- "Num Chiral Bonds" returns the number of chiral bonds in the molecule.

17.7.23 DatatableMolStringFunction

```
std::string DatatableMolStringFunction(const std::string &datatable, int row,
                                       const std::string &func)
```

Calculates molecular data that returns a string value. *datatable* identifies the datatable containing the molecule and row is the datatable row with the molecule.

The function *func* to compute is one of:

- "molformula" the molecular formula for the molecule.

17.7.24 DatatableNumRows

```
int DatatableNumRows(std::string datatable)
```

Return the number of rows for the datatable named *datatable*.

17.7.25 DatatableSetData

```
void DatatableSetData(const std::string &datatable, unsigned int row,
                      std::string header, std::string value, bool update=true)
```

Set the cell data for the datatable named *datatable* at *row* for the column named *header* to the string value *value*.

If update is True, immediately update the datatable. Set this to False if you are updating a bunch of data and then call DatatableUpdateContents on this datatable.

17.7.26 DatatableSetExpression

```
void DatatableSetExpression(const std::string &datatable,
                            const std::string &col, const std::string &expr)
```

A datatable expression defines an arbitrary piece of python code to call that generates data to be displayed in the datatable.

This function creates a new column named *col* for the specified datatable using the function defined by *expr*.

This expression may assume that a local variable *ROW* is assigned to the row currently being evaluated. e.g.

DatatableData(DatatableCurrent(), ROW, foo)

returns the string representation of the data in column *foo* for the current datatable which, for this function, is always *datatable*.

17.7.27 DatatableSetRowData

```
void DatatableSetRowData(const std::string &datatable, unsigned int row,
                         std::vector<std::string> headers,
                         std::vector<std::string> rows, bool update=true)
```

Set data in the datatable named *datatable*. *row* is the row index to set. *headers* defines the names of the columns to set and *rows* is the string representation of the data.

Note: headers[i] should be the name of the column for row[i].

The update parameter is no longer used and is kept for backwards compatibility.

17.7.28 SpreadsheetAddColumn

```
void SpreadsheetAddColumn(std::string spreadsheet, std::string column,
                          bool genericData=false)
```

Add a column to spreadsheet *spreadsheet* with the name *columnName*. If *columnName* already exists in the spreadsheet, it will be ignored.

For the "Atoms" and "Residues" spreadsheets, only certain column names are allowed, and the values in these columns cannot be changed.

For the "Atoms" spreadsheet, the valid column names are:

- x
- y
- z
- Radius
- Element
- Residue_Idx
- Name
- oe_atom_Formal_Charge
- oe_atom_Partial_Charge
- oe_atom_Idx
- oe_atom_Map_Idx
- oe_atom_Isotope
- oe_atom_Hyb
- oe_atom_Implicit_H_Count
- oe_atom_Explicit_H_Count
- oe_atom_Int_Type
- oe_atom_Type
- oe_atom_Aromatic
- oe_atom_Chiral
- oe_atom_In_Ring
- oe_residue_Name
- oe_residue_Occupancy
- oe_residue_BFactor
- oe_residue_Number
- oe_residue_Serial_Number
- oe_residue_Model_Number
- oe_residue_Fragment_Number
- oe_residue_Secondary_Structure
- oe_residue_Alternate_Location

- oe_residue_Chain_ID
- oe_residue_IsHetAtom

The *oe_residue_* and *oe_atom_* prefixes are not displayed in the spreadsheet headers, and are omitted when referring to these columns in other functions, such as SpreadsheetData and SpreadsheetRemoveColumn.

For the "Residues" spreadsheet, the valid column names are:

- Residue
- Average BFactor
- Min Occupancy
- Max Occupancy
- Alt Group
- Num AltConfs

17.7.29 SpreadsheetColumnColorerSet

```
void SpreadsheetColumnColorerSet(const std::string &spreadsheet,
                                 const std::string &column,
                                 const std::string &colorer, double min,
                                 double max)
```

Sets a colorer for the specified column in the specified spreadsheet. Valid values for the *colorer* parameter include:

- "redtoblue"
- "bluetored"
- "rainbow"
- "reverse rainbow"
- "redyellowblue"
- "greytogreen"

17.7.30 SpreadsheetColumnController

```
void SpreadsheetColumnController()
```

Open the spreadsheet column controller dialog. The column controls allows controlling the visibility and other aspects for the current spreadsheets column.

17.7.31 SpreadsheetColumnFontGet

```
std::string SpreadsheetColumnFontGet(const std::string &spreadsheet,
                                     const std::string &column)
```

Returns the font to be used when rendering text in the specified column of the specified spreadsheet.

17.7.32 SpreadsheetColumnFontSet

```
void SpreadsheetColumnFontSet(const std::string &spreadsheet,
                              const std::string &column,
                              const std::string &font)
```

Sets the font to be used when rendering text in the specified column of the specified spreadsheet.

17.7.33 SpreadsheetColumnReadonly

```
bool SpreadsheetColumnReadonly(std::string spreadsheet, unsigned int col)
bool SpreadsheetColumnReadonly(std::string spreadsheet, unsigned int col,
                               bool readonly)
```

Set a spreadsheet column to be read-only.

Without a readOnly value, returns the current readonly state for column *col* in *spreadsheet*.

With a readOnly specified, column *col* in the spreadsheet named *spreadsheet* is set to value set by readOnly.

17.7.34 SpreadsheetColumnSigFigGet

```
int SpreadsheetColumnSigFigGet(const std::string &spreadsheet,
                               const std::string &column)
```

Returns the number of significant figures used to display numerical values data in the specified column.

17.7.35 SpreadsheetColumnSigFigSet

```
void SpreadsheetColumnSigFigSet(const std::string &spreadsheet,
                                const std::string &column, int digits)
```

Sets the number of significant figures used to display numerical values data in the specified column.

17.7.36 SpreadsheetCommitChanges

```
void SpreadsheetCommitChanges(const std::string &spreadsheet)
```

Commit any unsaved changes to the spreadsheet named *spreadsheet*.

Normally, this is done automatically behind the scenes but sometimes is necessary to keep the data in the spreadsheet synchronized with the molecules and data in the repository.

17.7.37 SpreadsheetCopy

```
std::string SpreadsheetCopy(const std::string &spreadsheet)
```

Copies to the clipboard and returns the currently selected set of cells in the specified spreadsheet.

17.7.38 SpreadsheetCreateColumnExpression

```
void SpreadsheetCreateColumnExpression()
```

Open the dialog to create a new column expression for the current spreadsheet. A column expression is a user-defined function that populates a column with arbitrary data.

17.7.39 SpreadsheetCreateFilter

```
void SpreadsheetCreateFilter()
```

Open the dialog to generate a new view of the spreadsheet with all rows filtered by an arbitrary python expression.

17.7.40 SpreadsheetCurrentGet

```
std::string SpreadsheetCurrentGet()
```

Returns the name of the current spreadsheet. e.g. 'Molecules'.

17.7.41 SpreadsheetCurrentSet

```
bool SpreadsheetCurrentSet(const std::string &spreadsheet)
```

Sets the current spreadsheet.

17.7.42 SpreadsheetData

```
std::string SpreadsheetData(const std::string &spreadsheet, unsigned int row,
                            std::string header)
```

Return the string representation of the data in the spreadsheet named *spreadsheet* for the row *row* and the column named *header*.

17.7.43 SpreadsheetDeleteColumn

```
void SpreadsheetDeleteColumn(const std::string &column)
```

Delete the column named *column*. Note that this deletes *column* from all spreadsheets.

17.7.44 SpreadsheetEditableGet

```
bool SpreadsheetEditableGet(const std::string &ss)
```

Returns True if *spreadsheet* is editable.

17.7.45 SpreadsheetEditableSet

```
void SpreadsheetEditableSet(const std::string &ss, bool val)
```

Sets *spreadsheet* to be editable if *val* is True, otherwise sets *spreadsheet* to be read-only.

17.7.46 SpreadsheetFilter

```
void SpreadsheetFilter(const std::string &spreadsheet, std::string filter,
                       std::string name, bool staticView)
void SpreadsheetFilter(const std::string &spreadsheet, const OEDataFilter &p,
                       const std::string &name, bool staticview,
                       bool permaFilter=false)
```

Create a new spreadsheet view from the spreadsheet named *spreadsheet* using the filter defined in *filter*. The new view is named *name*. *filter* is an arbitrary python expression that is applied to every row in the spreadsheet.

This expression may assume that a local variable *ROW* is assigned to the row currently being evaluated. e.g.

SpreadsheetData(SpreadSheetCurrent(), ROW, foo)

returns the string representation of the data in column foo for the current spreadsheet. Note: the current spreadsheet may not be *spreadsheet*.

17.7.47 SpreadsheetFromList

```
void SpreadsheetFromList(unsigned int listid, const std::string &name)
```

Generate a filtered spreadsheet with name *name* from the list *listid*.

If *name* is already in the spreadsheet, it will be renamed name1, name2 and so on.

17.7.48 SpreadsheetGetColumn

```
std::vector<std::string> SpreadsheetGetColumn(const std::string &spreadsheet,
                                              unsigned int index)
std::vector<std::string> SpreadsheetGetColumn(const std::string &spreadsheet,
                                              const std::string &name)
```

Returns string representations of the entire column for the spreadsheet named *spreadsheet* for column *name* or, alternatively, for column index by *index*.

17.7.49 SpreadsheetGetCurrentRow

```
int SpreadsheetGetCurrentRow(std::string spreadsheet)
```

Returns the row index for the first active or selected row for the spreadsheet named *spreadsheet*.

17.7.50 SpreadsheetGetIDForRow

```
unsigned int SpreadsheetGetIDForRow(std::string spreadsheet, unsigned int row)
```

Returns the repository id for the first active or selected row for the spreadsheet named *spreadsheet*.

17.7.51 SpreadsheetGetImageStreamAtRow

```
bool SpreadsheetGetImageStreamAtRow(const std::string &spreadsheet,
                                    const std::string &filename, int row,
                                    int width=256, int height=256)
```

Write an image file to the file specified by *filename* for the spreadsheet named *spreadsheet* at the row specified by *row*. The size of the image is specified by *width* and *height*.

17.7.52 SpreadsheetGetKeyForRow

```
OEPropDB::OEKey SpreadsheetGetKeyForRow(std::string spreadsheet,
                                        unsigned int row)
```

Returns the OEKey value for the first active or selected row for the spreadsheet named *spreadsheet*.

17.7.53 SpreadsheetGetNumRows

```
int SpreadsheetGetNumRows(std::string spreadsheet)
```

Return the number of rows for the spreadsheet named *spreadsheet*.

17.7.54 SpreadsheetGetRowForKey

```
unsigned int SpreadsheetGetRowForKey(std::string spreadsheet, OEPropDB::OEKey)
```

Returns the row that is generated from OEKey *key*. Returns "4294967295L" if *key* is not in the spreadsheet.

17.7.55 SpreadsheetGetSpreadsheets

```
std::vector<std::string> SpreadsheetGetSpreadsheets()
```

Return a list of the names for all the currently available spreadsheets.

17.7.56 SpreadsheetHeaders

```
std::vector<std::string> SpreadsheetHeaders(const std::string &spreadsheet)
```

Return a list of all the names of all the headers for the spreadsheet named *spreadsheet*.

17.7.57 SpreadsheetHideColumn

```
void SpreadsheetHideColumn(const std::string &spreadsheet,
                           const std::string &name)
```

Hide the column named *name* for the spreadsheet named *spreadsheet*.

17.7.58 SpreadsheetHideTab

```
bool SpreadsheetHideTab(std::string name)
```

Hide the named spreadsheet from the spreadsheet view.

17.7.59 SpreadsheetImport

```
int SpreadsheetImport(const VFSpreadSplitter &splitter,
                      const std::string &filename,
                      unsigned int matchBy=ImportSpreadsheet::ImportByOrder,
                      const std::string &matchList="",
                      unsigned int importBy=ImportSpreadsheet::ImportAsIs,
                      const std::string &importColumnOrFunction="")
```

Imports the specified file into the spreadsheet.

17.7.60 SpreadsheetLingoSimSort

```
void SpreadsheetLingoSimSort(const std::string &spreadsheet, unsigned int row)
```

Sorts the specified spreadsheet according to Lingo similarity to the molecule in the specified row.

17.7.61 SpreadsheetLoadFilter

```
void SpreadsheetLoadFilter(const std::string &spreadsheet,
                           const OEDataFilter &filter)
void SpreadsheetLoadFilter(const std::string &spreadsheet,
                           const OEDataFilter &filter, bool wasStatic,
                           bool isStatic)
```

Creates a new spreadsheet using the specified filter.

17.7.62 SpreadsheetMolNumberFunction

```
double SpreadsheetMolNumberFunction(const std::string &spreadsheet, int row,
                                    const std::string &func)
```

Calculates molecular data that returns a numeric value.

spreadsheet identifies the spreadsheet containing the molecule and row is the spreadsheet row with the molecule.

The function *func* to compute is one of:

- "mw" Molecular weight.
- "Num Atoms" Number of atoms in the molecule.
- "Num Bonds" number of bonds in the molecule.
- **"Carbon-Hetero ratio" the ratio of carbons to hetero atoms in the molecule.**
  Returns -1 if there are no carbons.
- "Energy" The molecular energy of the molecule as specified in the input file.
- "Actual Charge" The sum of the partial charges on all atoms as specified in the input file.
- "Formal Charge" The sum of the formal charges on all atoms.
- "Halide Count" The number of halogen atoms in the molecule.
- "Num Carbons" The number of carbon atoms in the molecule.
- "Num Formal Charges" The number of more atoms with a specified formal charge.
- "Num Heavy Atoms" The number of heavy atoms (non hydrogen) in the molecule.
- "Num Hetero Atoms" The number of hetero atoms in the molecule.
- "Num Hydrogens" The number of hydrogen atoms in the molecule.
- "Num Rigid Bonds" The number of rigid bonds in the molecule.
- "Nom Rotatable Bonds" The number of rotatable bonds in the molecule.
- "Num Chiral Atoms" returns the number of chiral atoms in the molecule.
- "Num Chiral Bonds" returns the number of chiral bonds in the molecule.

17.7.63 SpreadsheetMolStringFunction

```
std::string SpreadsheetMolStringFunction(const std::string &spreadsheet,
                                         int row, const std::string &func)
```

Calculate molecular data that returns a string value.

spreadsheet identifies the spreadsheet containing the molecule and row is the spreadsheet row with the molecule.

The function *func* to compute is one of:

- 'molformula' the molecular formula for the molecule.

17.7.64 SpreadsheetMoveColumn

```
void SpreadsheetMoveColumn(const std::string &spreadsheet,
                           const std::string &columnName, int position)
```

Moves the specified column to the specified position.

17.7.65 SpreadsheetNumRows

```
int SpreadsheetNumRows(std::string spreadsheet)
```

Return the number of rows for the spreadsheet named *spreadsheet*.

17.7.66 SpreadsheetPromptColumnExpression

```
std::string SpreadsheetPromptColumnExpression()
```

This function opens a dialog and returns the expression to evaluate to add the specified column expression.

17.7.67 SpreadsheetPromptExport

```
std::string SpreadsheetPromptExport(const std::string &filename)
```

This function opens a dialog and returns the expression to export the specified spreadsheet and columns.

17.7.68 SpreadsheetPromptFilter

```
std::string SpreadsheetPromptFilter()
```

This function opens a dialog and returns the expression to evaluate to filter the specified spreadsheet.

17.7.69 SpreadsheetPromptFormat

```
std::string SpreadsheetPromptFormat()
```

This function opens a dialog which allows the user to specify the formatting of the spreadsheet.

17.7.70 SpreadsheetPromptGraphemeOpts

```
std::string SpreadsheetPromptGraphemeOpts()
```

This function opens a dialog which allows the user to specify how the 2D depiction is rendered using OpenEye's **Grapheme** toolkit to apply a property map of computed properties or even user defined properties which are tagged to the atoms using generic data.

17.7.71 SpreadsheetPromptImport

```
std::string SpreadsheetPromptImport(const std::string &filename)
```

This function opens a dialog and returns the expression to import spreadsheet *filename*.

17.7.72 SpreadsheetPromptSort

```
std::string SpreadsheetPromptSort()
```

This function opens a dialog and returns the expression to evaluate to sort the specified spreadsheet and columns.

17.7.73 SpreadsheetRemoveTab

```
bool SpreadsheetRemoveTab(std::string name)
```

Remove the spreadsheet with name *name*. The base "Molecules" spreadsheet cannot be removed.

17.7.74 SpreadsheetRowHeightGet

```
int SpreadsheetRowHeightGet(const std::string &spreadsheet)
```

Returns the row height for the specified spreadsheet.

17.7.75 SpreadsheetRowHeightSet

```
void SpreadsheetRowHeightSet(const std::string &spreadsheet, int height)
```

Sets the default row height for the specified spreadsheet.

17.7.76 SpreadsheetSetData

```
void SpreadsheetSetData(const std::string &spreadsheet, unsigned int row,
                        std::string header, std::string value, bool update=true)
```

Set the cell data for the spreadsheet named *spreadsheet* at *row* for the column named *header* to the string value *value*.

If update is true, immediately update the spreadsheet. Set this to false if you are updating a bunch of data and then call *SpreadsheetUpdateContents* on the spreadsheet.

17.7.77 SpreadsheetSetExpression

```
void SpreadsheetSetExpression(const std::string &spreadsheet,
                              const std::string &col, const std::string &expr)
```

A spreadsheet expression defines an arbitrary piece of python code to call that generates data to be displayed in the spreadsheet.

SpreadsheetSetExpression creates a new column named *col* for the spreadsheet named *spreadsheet* using the function defined by *expr*.

This expression may assume that a local variable *ROW* is assigned to the row currently being evaluated. e.g.

SpreadsheetData(SpreadSheetCurrent(), ROW, foo)

returns the string representation of the data in column *foo* for the current spreadsheet which, for this function, is always *spreadsheet*.

17.7.78 SpreadsheetSetRowData

```
void SpreadsheetSetRowData(const std::string &spreadsheet, unsigned int row,
                           std::vector<std::string> headers,
                           std::vector<std::string> rows, bool update=true)
```

Set data in the spreadsheet named *spreadsheet*. *row* is the row index to set. *headers* defines the names of the columns to set and *rows* is the string representation of the data.

Note: headers[i] should be the name of the column for row[i].

If update is true, immediately update the spreadsheet. Set this to false if you are updating a bunch of data and then call SpreadsheetUpdateContents(ss).

17.7.79 SpreadsheetShowAllColumns

```
void SpreadsheetShowAllColumns(const std::string &spreadsheet)
```

Make all columns visible for the spreadsheet named *spreadsheet*.

17.7.80 SpreadsheetShowAllTabs

```
void SpreadsheetShowAllTabs()
```

Make all spreadsheets visible.

17.7.81 SpreadsheetShowColumn

```
void SpreadsheetShowColumn(const std::string &spreadsheet, int section,
                           bool visible=true)
void SpreadsheetShowColumn(const std::string &spreadsheet,
                           const std::string &name, bool visible=true)
```

Make column *name* visible for the spreadsheet named *spreadsheet*.

17.7.82 SpreadsheetShowStats

```
void SpreadsheetShowStats(const std::string &spreadsheet, bool show)
```

If *show* is true, show the column statistics for the spreadsheet named *spreadsheet*. Otherwise, hide the statistics.

17.7.83 SpreadsheetShowStatsGet

```
bool SpreadsheetShowStatsGet(const std::string &spreadsheet)
```

Returns True if *spreadsheets* has the statistics window shown, False otherwise.

17.7.84 SpreadsheetShowStatsSet

```
void SpreadsheetShowStatsSet(const std::string &spreadsheet, bool show)
```

If *show* is True, the statistics window for *spreadsheet* is shown. Otherwise it is hidden.

17.7.85 SpreadsheetShowTab

```
bool SpreadsheetShowTab(std::string name)
```

Ensure that *spreadsheet* is shown as a spreadsheet selection in the spreadsheet window.

17.7.86 SpreadsheetSort

```
void SpreadsheetSort(const std::string &spreadsheet,
                     const std::vector<std::string> &columns,
                     const std::vector<int> &directions,
                     bool moveToFirst=true)
```

Sort *spreadsheet* by the specified columns and directions. If a direction is 1 then the column is ascending, if a direction is 2 the column is descending.

Example:

```
SpreadsheetSort( "Molecules", ["target", "IC50"], [1,2] )

target | IC50
cox2   | 1.60
cox2   | 1.40
cox1   | 1.80
cox1   | 1.67

If *moveToFirst* is True then the sorted columns will be placed first in the spreadsheet.

This would sort the molecules by target ascending and then by IC50 descending
```

17.8 Deprecated Functions

The functions detailed in this section have been deprecated. They are still available for use, but are scheduled to be removed from the API in a future version. Most of these functions have been deprecated in order to create a more consistently named API, but some have been deprecated as the functionality they provided is no longer available, obsolete, or unnecessary. If an alternate function is available, it will be referenced in the documentation below.

17.8.1 ContourCurrentGridGet

```
unsigned int ContourCurrentGridGet()
```

This function has been deprecated.

17.8.2 ContourCurrentGridSet

```
unsigned int ContourCurrentGridSet(unsigned int id)
```

This function has been deprecated.

17.8.3 CreateAngleMonitor

```
unsigned int CreateAngleMonitor(const OEPropDB::OEKey &k1,
                                const OEPropDB::OEKey &k2,
                                const OEPropDB::OEKey &k3)
```

This function has been deprecated. Please use *MonitorAngleCreate* instead.

17.8.4 CreateDistanceMonitor

```
unsigned int CreateDistanceMonitor(const OEPropDB::OEKey &k1,
                                   const OEPropDB::OEKey &k2)
```

This function has been deprecated. Please use *MonitorDistanceCreate* instead.

17.8.5 CreateSphereMonitor

```
unsigned int CreateSphereMonitor(const OEPropDB::OEKey &k,
                                 const std::string &name,
                                 const OESystem::OEColor &c, float rad=0.0f)
unsigned int CreateSphereMonitor(const OEPropDB::OEKey &k,
                                 const std::string &name, float x, float y,
                                 float z, float rad, const OESystem::OEColor &c)
```

This function has been deprecated. Please use *MonitorSphereCreate* instead.

17.8.6 CreateTorsionMonitor

```
unsigned int CreateTorsionMonitor(const OEPropDB::OEKey &k1,
                                  const OEPropDB::OEKey &k2,
                                  const OEPropDB::OEKey &k3,
                                  const OEPropDB::OEKey &k4)
```

This function has been deprecated. Please use *MonitorTorsionCreate* instead.

17.8.7 CustomViewDocking

```
void CustomViewDocking()
void CustomViewDocking(const std::string &protein, const std::string &reference,
                       const std::string &results="")
void CustomViewDocking(const std::string &protein, const std::string &reference,
                       const std::string &results, bool showRibbons,
                       bool showLabels, bool showHBonds, bool showSurface,
                       float residueWithin, float surfaceWithin,
                       const std::string &surfaceColor)
```

This function has been deprecated. Please use *CustomViewFRED* instead.

17.8.8 DatatableCurrent

```
std::string DatatableCurrent()
```

This function has been deprecated. Please use *DatatableCurrentGet* instead.

17.8.9 DeleteAngleMonitor

```
void DeleteAngleMonitor(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                        const OEPropDB::OEKey &k3)
```

This function has been deprecated. Please use *MonitorAngleDelete* instead.

17.8.10 DeleteDistanceMonitor

```
void DeleteDistanceMonitor(const OEPropDB::OEKey &k1,
                           const OEPropDB::OEKey &k2)
```

This function has been deprecated. Please use *MonitorDistanceDelete* instead.

17.8.11 DeleteTorsionMonitor

```
void DeleteTorsionMonitor(const OEPropDB::OEKey &k1, const OEPropDB::OEKey &k2,
                          const OEPropDB::OEKey &k3, const OEPropDB::OEKey &k4)
```

This function has been deprecated. Please use *MonitorTorsionDelete* instead.

17.8.12 DeleteVisibleMonitors

```
void DeleteVisibleMonitors()
```

This function has been deprecated. Please use *MonitorsVisibleDelete* instead.

17.8.13 ExistsAngleMonitor

```
unsigned int ExistsAngleMonitor(const OEPropDB::OEKey &k1,
                                const OEPropDB::OEKey &k2,
                                const OEPropDB::OEKey &k3)
```

This function has been deprecated. Please use *MonitorAngleExists* instead.

17.8.14 ExistsDistanceMonitor

```
unsigned int ExistsDistanceMonitor(const OEPropDB::OEKey &k1,
                                   const OEPropDB::OEKey &k2)
```

This function has been deprecated. Please use *MonitorDistanceExists* instead.

17.8.15 ExistsTorsionMonitor

```
unsigned int ExistsTorsionMonitor(const OEPropDB::OEKey &k1,
                                  const OEPropDB::OEKey &k2,
                                  const OEPropDB::OEKey &k3,
                                  const OEPropDB::OEKey &k4)
```

This function has been deprecated. Please use *MonitorTorsionExists* instead.

17.8.16 GetAtomsByScope

```
std::vector<OEPropDB::OEKey> GetAtomsByScope(unsigned int scope)
```

This function has been deprecated. Please use *GetAtomsScoped* instead.

17.8.17 GetBondsByScope

```
std::vector<OEPropDB::OEKey> GetBondsByScope(unsigned int scope)
```

This function has been deprecated. Please use *GetBondsScoped* instead.

17.8.18 GetContoursByScope

```
std::vector<OEPropDB::OEKey> GetContoursByScope(unsigned int scope)
```

This function has been deprecated. Please use *GetContoursScoped* instead.

17.8.19 GetGridsByScope

```
std::vector<OEPropDB::OEKey> GetGridsByScope(unsigned int scope)
```

This function has been deprecated. Please use *GetGridsScoped* instead.

17.8.20 GetKey

```
OEPropDB::OEKey GetKey(const OESystem::OEBase &base)
OEPropDB::OEKey GetKey(const OESystem::OEBase &base,
                       const OEPropDB::OEKey &parentKey)
```

This function has been deprecated. Please use *KeyGet* instead.

17.8.21 GetKeyType

```
std::string GetKeyType(const OEPropDB::OEKey &key)
```

This function has been deprecated. Please use *KeyTypeGet* instead.

17.8.22 GetKeysByScope

```
std::vector<OEPropDB::OEKey> GetKeysByScope(unsigned int scope,
                                            unsigned int type)
```

This function has been deprecated. Please use *GetScoped* instead.

17.8.23 GetKeysForID

```
std::vector<OEPropDB::OEKey> GetKeysForID(unsigned int id)
```

This function has been deprecated. Please use *KeysGet* instead.

17.8.24 GetMoleculesByScope

```
std::vector<OEPropDB::OEKey> GetMoleculesByScope(unsigned int scope)
```

This function has been deprecated. Please use *GetMoleculesScoped* instead.

17.8.25 GetName

```
std::string GetName(const OEPropDB::OEKey &key)
```

This function has been deprecated. Please use *NameGet* instead.

17.8.26 GetPropertyType

```
unsigned int GetPropertyType(const std::string &type)
```

This function has been deprecated. Please use *PropertyTypeGet* instead.

17.8.27 GetSurfacesByScope

```
std::vector<OEPropDB::OEKey> GetSurfacesByScope(unsigned int scope)
```

This function has been deprecated. Please use *GetSurfacesScoped* instead.

17.8.28 GotoStylePage

```
void GotoStylePage(const std::string &page)
```

This function has been deprecated.

17.8.29 InitializeObject

```
void InitializeObject(unsigned int id)
```

This function has been deprecated. Please use *Initialize* instead.

17.8.30 InterpreterInteractiveGet

```
bool InterpreterInteractiveGet()
```

This function has been deprecated.

17.8.31 InterpreterInteractiveSet

```
void InterpreterInteractiveSet(bool interactive)
```

This function has been deprecated.

17.8.32 MarkObjectsByScope

```
bool MarkObjectsByScope(unsigned int scope)
```

This function has been deprecated. Please use *MarkScoped* instead.

17.8.33 MarkState

```
int MarkState()
```

This function has been deprecated. Please use *StateMark* instead.

17.8.34 MenuAddLabel

```
std::string MenuAddLabel(const std::string &menu, const std::string &name,
                         bool last)
```

This function has been deprecated.

17.8.35 MenuUseTearoffsGet

```
bool MenuUseTearoffsGet()
```

This function has been deprecated.

17.8.36 MenuUseTearoffsSet

```
bool MenuUseTearoffsSet(bool b)
```

This function has been deprecated.

17.8.37 OEGetKey

```
OEPropDB::OEKey OEGetKey(const OESystem::OEBase &b)
OEPropDB::OEKey OEGetKey(const OESystem::OEBase &b, const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *KeyGet* instead.

17.8.38 OEKeyToID

```
unsigned int OEKeyToID(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *KeyIDGet* instead.

17.8.39 OEKeyToLabelString

```
std::string OEKeyToLabelString(const OEPropDB::OEKey &k, bool full=false,
                               bool coords=true)
```

This function has been deprecated. Please use *LabelGet* instead.

17.8.40 OEKeyToParentID

```
unsigned int OEKeyToParentID(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *KeyParentIDGet* instead.

17.8.41 OEKeyToSourceID

```
unsigned int OEKeyToSourceID(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *KeySourceIDGet* instead.

17.8.42 OEPyClearActive

```
void OEPyClearActive()
```

This function has been deprecated. Please use *ClearActive* instead.

17.8.43 OEPyClearLocked

```
void OEPyClearLocked()
```

This function has been deprecated. Please use *ClearLocked* instead.

17.8.44 OEPyClearMarked

```
void OEPyClearMarked()
```

This function has been deprecated. Please use *ClearMarked* instead.

17.8.45 OEPyClearSelected

```
void OEPyClearSelected()
```

This function has been deprecated. Please use ClearSelected instead.

17.8.46 OEPyClearVisible

```
void OEPyClearVisible()
```

This function has been deprecated. Please use *ClearVisible* instead.

17.8.47 OEPyGetActive

```
OEPropDB::OEKey OEPyGetActive()
```

This function has been deprecated. Please use *ActiveKey* instead.

17.8.48 OEPyGetIDForKey

```
unsigned int OEPyGetIDForKey(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *KeyIDGet* instead.

17.8.49 OEPyGetSelectedAtoms

```
std::vector<OEPropDB::OEKey> OEPyGetSelectedAtoms()
```

This function has been deprecated. Please use *GetSelectedAtoms* instead.

17.8.50 OEPyHide

```
bool OEPyHide(unsigned int scope, bool value)
```

This function has been deprecated. Please use *HideScoped* instead.

17.8.51 OEPyHideNone

```
bool OEPyHideNone(unsigned int scope)
```

This function has been deprecated. Please use HideNonScoped instead.

17.8.52 OEPyHideOthers

```
bool OEPyHideOthers(const std::vector<OEPropDB::OEKey> &keys, bool hide)
```

This function has been deprecated. Please use *HideOthers* instead.

17.8.53 OEPyIsActive

```
bool OEPyIsActive(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *IsActive* instead.

17.8.54 OEPyIsLocked

```
bool OEPyIsLocked(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *IsLocked* instead.

17.8.55 OEPyIsMarked

```
bool OEPyIsMarked(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *IsMarked* instead.

17.8.56 OEPyIsSelected

```
bool OEPyIsSelected(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *IsSelected* instead.

17.8.57 OEPyIsTrulyVisible

```
bool OEPyIsTrulyVisible(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *IsTrulyVisible* instead.

17.8.58 OEPyIsVisible

```
bool OEPyIsVisible(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *IsVisible* instead.

17.8.59 OEPySetActive

```
bool OEPySetActive(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *Active* instead.

17.8.60 OEPySetLocked

```
bool OEPySetLocked(const OEPropDB::OEKey &k, bool state)
bool OEPySetLocked(const std::vector<OEPropDB::OEKey> &keys, bool state)
```

This function has been deprecated. Please use *Lock* instead.

17.8.61 OEPySetMarked

```
bool OEPySetMarked(const OEPropDB::OEKey &k, bool state)
bool OEPySetMarked(const std::vector<OEPropDB::OEKey> &keys, bool state)
```

This function has been deprecated. Please use *Mark* instead.

17.8.62 OEPySetSelected

```
bool OEPySetSelected(const OEPropDB::OEKey &k, bool state)
bool OEPySetSelected(const std::vector<OEPropDB::OEKey> &keys, bool state)
```

This function has been deprecated. Please use *Select* instead.

17.8.63 OEPySetVisible

```
bool OEPySetVisible(const OEPropDB::OEKey &k, bool state)
bool OEPySetVisible(const std::vector<OEPropDB::OEKey> &keys, bool state)
```

This function has been deprecated. Please use *Visible* instead.

17.8.64 ObjectNameGet

```
std::string ObjectNameGet(unsigned int id)
std::string ObjectNameGet(const OEPropDB::OEKey &k)
```

This function has been deprecated. Please use *NameGet* instead.

17.8.65 ObjectNameSet

```
void ObjectNameSet(unsigned int id, const std::string &name)
void ObjectNameSet(const OEPropDB::OEKey &k, const std::string &name)
```

This function has been deprecated. Please use *NameSet* instead.

17.8.66 ObservableOEKey

```
OEGUI::OEObservableBase<OEPropDB::OEKey> &
  ObservableOEKey(const std::string &name, bool create=false)
```

This function has been deprecated. Please use *ObservableKey* instead.

17.8.67 PopState

```
int PopState(bool load=true)
```

This function has been deprecated. Please use *StatePop* instead.

17.8.68 PopupAddLabel

```
std::string PopupAddLabel(const std::string &label, bool once=false)
```

This function has been deprecated.

17.8.69 ResponseVectMulti

```
void ResponseVectMulti(const std::string &s,
                       const std::vector<OEInterpreter::OEMultiTypeVar> &var,
                       bool b=false)
```

This function has been deprecated. Please use *PromptResponseVectMulti* instead.

17.8.70 SDataGet

```
std::string SDataGet(unsigned int id, const std::string &tag)
std::string SDataGet(const OEPropDB::OEKey &k, const std::string &tag)
```

This function has been deprecated. Please use *SDDataGet* instead.

17.8.71 SDataHas

```
bool SDataHas(unsigned int id, const std::string &tag)
bool SDataHas(const OEPropDB::OEKey &k, const std::string &tag)
```

This function has been deprecated. Please use *SDDataHas* instead.

17.8.72 SDataSet

```
void SDataSet(unsigned int id, const std::string &tag, const std::string &v)
void SDataSet(const OEPropDB::OEKey &k, const std::string &tag,
              const std::string &v)
```

This function has been deprecated. Please use *SDDataSet* instead.

17.8.73 ScratchList

```
std::vector<OEPropDB::OEKey> ScratchList()
```

This function has been deprecated. Please use *ScratchGet* instead.

17.8.74 SpreadsheetCurrent

```
std::string SpreadsheetCurrent()
```

This function has been deprecated, please use *SpreadsheetCurrentGet* instead.

17.8.75 SpreadsheetDisableUpdates

```
void SpreadsheetDisableUpdates(const std::string &spreadsheet)
```

This function has been deprecated.

17.8.76 SpreadsheetEnableUpdates

```
void SpreadsheetEnableUpdates(const std::string &spreadsheet)
```

This function has been deprecated.

17.8.77 SpreadsheetMarkHighlighted

```
void SpreadsheetMarkHighlighted(const std::string &spreadsheet, bool marked)
```

This function has been deprecated.

17.8.78 SpreadsheetUpdateContents

```
void SpreadsheetUpdateContents(const std::string &spreadsheet)
```

This function has been deprecated.

17.8.79 SurfacePotentialColorAuto

```
void SurfacePotentialColorAuto()
```

This function has been deprecated.

17.8.80 SurfacePotentialColorValuesSet

```
void SurfacePotentialColorValuesSet(float min, float mid, float max)
```

This function has been deprecated.

17.8.81 VFCopyFile

```
bool VFCopyFile(const std::string &src, const std::string &dst)
```

This function has been deprecated.

17.8.82 VFGetMol

```
bool VFGetMol(OEChem::OEMCMolBase &m, unsigned int id)
bool VFGetMol(OEChem::OEMCMolBase &m, const OEPropDB::OEKey &key)
```

This function has been deprecated. Please use *MoleculeGet* instead.

CHAPTER

EIGHTEEN

RELEASE HISTORY

18.1 VIDA 5.0.7

18.1.1 New features

18.1.2 Major bug fixes

18.1.3 Minor bug fixes

18.2 VIDA 5.0.6

Minor internal improvements have been made.

18.3 VIDA 5.0.5

Minor internal improvements have been made.

18.4 VIDA 5.0.4

Minor internal improvements have been made.

18.5 VIDA 5.0.3

Fall 2022

Minor internal improvements have been made.

18.6 VIDA 5.0.2

Spring 2022

18.6.1 New Features

- `mcr` files are now a supported file type within VIDA.
- The ability to display in hardware 3D stereoscopic via OpenGL Quad Buffering on Windows with compatible graphics devices has been restored.

18.6.2 Minor Bug Fixes

- Reduced many warnings generated in the Scripting Window or in the terminal from which VIDA was launched.

18.7 VIDA 5.0.1

Fall 2021

- Minor internal improvements have been made.

18.8 VIDA 5.0.0

July 2021

18.8.1 New Features

- VIDA has been updated and ships with the current toolkit release 2021.1.
- VIDA has been updated to use Python 3.
- VIDA has been updated to use Qt 5. In addition we have switched for PyQt to PySide2, providing full access to the Qt library via Python.
- VIDA extensions have been updated to Python 3 and Qt 5.
- VIDA now supports opening OEDesignUnits and recognized the OEDU file format. VIDA will also show OEReceptor information if stored on the OEDesignunit. OEDesignUnits will not be able to be saved in this release.
- VIDA now supports opening MMCIF files.
- File->Open Special->From PDB will now download an MMCIF file if a PDB file is not available for a particular PDB code.
- Several toolbar icons have been updated

18.8.2 Major Bug Fixes

- Improvements have been made regarding loading of state files

18.8.3 Removed features

- Support for hardware stereo has been removed
- The screenshot tool has been removed

18.9 VIDA 4.4.0

February 2018

18.9.1 New Features

- A new ribbon style has been introduced into **Spicoli TK** and is now being used to render ribbons within **VIDA**. The new style produces ribbons with a smoother appearance.
- Various improvements to the Builder/Sketcher behavior have been made, including:
  - closing the Sketcher window now prompts for *Save*, *Save as New*, *Discard*, or *Cancel*
  - closing the Builder now closes the Sketcher window

  - an additional *Save As New* option appears in the toolbar and Builder context menu
  - hitting *Return* now finishes adding typed-in molecules from the Sketcher

- An *Unhide Atoms* option was added to the *Select* Menu.
- Extensions can now be deactivated centrally by appending `-remove` to the extension's entry in a PYXS subscription file. Various other improvements to the Extension Manager behavior were also made.
- 3D renderings of `OEAnnotation` objects are now selectable.
- Several improvements have been made to the Scripting window, including to the *Paste* and *Drag/Drop* behavior.

18.9.2 Major Bug Fixes

- A bug that caused waters to not show as hydrogen bond acceptors has been fixed.
- An issue that caused bond torsions to rotate atoms on the wrong end of the bond has been fixed.
- A crash caused by excessive memory use while centering a large number of molecules in All Scope mode has been rectified.
- A bug that could sometimes cause a crash when changing the font size in the 3D Window has been fixed.
- A memory leak that occurred when switching screenshots or switching layouts has been fixed.
- A crash that could occur when adding large number of grids has been fixed.
- A crash that could occur when shift-selecting surfaces on proteins has been fixed.
- A crash that occurred when clicking the *List Window*'s *visibility* header button when a grid and many molecules were loaded has been fixed.
- Coloring proteins using the *Alternate Locations* coloring scheme no longer crashes the program when applied to molecules without alternate locations specified.
- A conflict between the Qt libraries installed with **VIDA** and the Qt libraries included in Kubuntu would cause a "Cannot mix incompatible Qt libraries" error message when starting **VIDA**. This error has been corrected by including some additional Qt libraries in the **VIDA** distribution.

18.9.3 Minor Bug Fixes

- The *Save* dialog had been performing very slowly when a large number of molecules were loaded. This has been fixed.
- Screenshots made from the 2D Viewer are no longer saved with a border.
- An issue that caused a complete molecular surface to appear during operations on molecules with partial surfaces has been fixed.
- A problem with synchronizing selections between atoms in the 2D/3D Viewers and atoms in the SMILES display in the 3D Viewer has been fixed.
- Aromaticity of certain molecules with tetravalent sulfur atoms had previously displayed inconsistently between the sketcher and the 3D Viewer. This has been fixed
- A bug that could cause electrostatic grids generated using AM1BCC charges to generate incorrect contours has been fixed.
- The default electrostatic surface grid spacing has been increased.
- In certain cases, rounding errors were causing incorrect sorting behavior for very large numbers in the spreadsheet. This was fixed by changing the internal representation for spreadsheet numbers from floats to doubles.
- Changing the application layout no longer hides visible extension windows.

- A problem with setting bookmark animation timings has been fixed.
- Minor fixes to the *Extensions Manager* dialog have been made.
- Behavior of the `AtomLabelSetScoped` function has been fixed to make long label-type arguments (e.g., "residue info (ca)") work as described in the documentation.
- A bug that made hyperlinks in the spreadsheet case insensitive has been fixed.
- Some geometry issues with the Attach and Fuse operations in build mode have been fixed.
- A *Donor* option had been missing from the *Select Within Focused* dialog. This has been fixed.
- A bug that affected the preferences for depicting superatoms in 2D renderings has been fixed. In addition, a new dialog for specifying which functional groups to render as superatoms has been added.
- The *Show Axes* preference under the 3D Display General preferences now displays the axes at the center of the 3D view.

18.9.4 Platform Support

- Support for 64-bit Windows has been added.
- 32-bit Windows is no longer supported.
- Support for Ubuntu 16 has been added.
- Support for RedHat v7 has been added.
- RedHat v5 is no longer supported.
- Support for Mac OS X 10.10 Yosemite has been added.
- Support for Mac OS X 10.11 El capitan has been added.
- Support for macOS 10.12 Sierra has been added.
- Support for macOS 10.13 High Sierra has been added.

18.9.5 Other

- Version 4.4.0 will be the last version of **VIDA** supporting Python 2.7. Future versions will be based on Python 3.

18.10 VIDA 4.3.0

August 2014

18.10.1 New Features

- "Drag and Drop" is now supported for rearranging items in the List Window.
- Advanced depiction options, including atom property maps from the **Grapheme TK**, substructure highlighting, and 2D structure alignment, are available for depiction in the 2D window and spreadsheet.
- Improved surface rendering using the new "atom splitting" feature from the **Spicoli TK**.
- Atom-based selection of surfaces.
- Atom and surface selection is now indicated using bright outlines instead of atom coloring.
- A new "Select" menu provides a wide range of ways to select atoms. The menu also allows creation and manipulation of named sets of atoms.

- A surface hydrophobicity coloring scheme based on atomic XLogP contributions was added.
- A new surface "curvature" coloring scheme was added that combines the previous "concavity" and "curvature" schemes.
- An additional **VIDA** extension for coloring molecules by SD data is included.
- An example "Cluster Browsing" script, for viewing clustered molecule data, is included.
- Many enhancements were made to the Extensions Manager dialog, to allow better handling of extension subscriptions. Other extension improvements include the ability to specify the order in which extensions are loaded, and the ability to use the VIDA_EXTENSIONS_URL environment variable to specify multiple auto-subscriptions.
- Molecular surfaces can now be created over sets of selected atoms, and can be regenerated via the Style Control.
- The **VIDA** spreadsheet can now include generic data stored on molecules and conformers.
- The List Window now allows clicking on column headers to show/hide, mark, and lock all items in the list.
- The builder now issues a warning (and opportunity to cancel) if a protein molecule has been put into Edit mode.
- Extension authoring was made easier by allowing local filenames in .pyx and .pyxs files.

18.10.2 Major Bug Fixes

- A crash from undoing conformer deletion has been fixed.
- A number of stability issues and strange window behavior under OS X 10.8+ have been resolved.
- A crash while creating C-alpha traces for some proteins was fixed.
- Several operations in the sketcher could create radicals unexpectedly. This has been corrected.

18.10.3 Minor Bug Fixes

- Fixed a warning message when opening .cdx files.
- A problem with state files not preserving child object visibilities was corrected.
- SMILES (.smi) files now contain stereochemical information by default.
- Installation of some extensions caused reading of large files to slow down significantly.
- The save dialog was very slow to initialize for large files.
- New lists created (e.g., from "Marked" molecules) now become the current active list.
- Switching to stereo during a telemodeling session would lose synchronization between the connected sessions.
- B-Factor labels can now be shown on any atoms, not just C-alpha atoms.
- Files opened via the command line now appear in the "Recents" list.
- Fixed an issue with extension updates requiring an extra **VIDA** restart to take effect.
- A problem with exiting fullscreen mode was fixed.
- A crash on exit when using certain extensions on Linux machines was fixed.
- Grids rendered from CCP4 files now use the statistics embedded in the file, rather than statistics generated within **VIDA**.

18.10.4 Platform Support

- Support for Ubuntu 14 was added.
- Support for RedHat v7 was added.
- Support for Mac OS X 10.8 and 10.9 was added.
- Mac OS X 10.6 is no longer supported.
- RedHat v4 is no longer supported.
- Windows XP is no longer supported.

18.10.5 Other

Vivant is being retired, and no longer ships with **VIDA**. Existing versions of **Vivant** will continue to function and remain compatible with **VIDA**'s current state files.

18.11 VIDA 4.2.0

September 2012

18.11.1 New Features

- A "telemodeling" feature has been introduced (as an included **VIDA** extension), and allows interactive sharing of **VIDA** sessions over a network.
- 3D graphics shading calculations have been redone, to provide much nicer looking rendering of 3D objects.
- Solid-rendered 3D objects now include edge outlines to improve clarity.
- **VIDA**'s spreadsheet has been updated with several new features:
  - Depictions now use the new **OEDepict TK**, and can be displayed in color.
  - Atom selection is reflected in the spreadsheets.
  - Structures from SMILES strings can now be displayed in spreadsheet cells. Just put the string "(smiles)" anywhere in the column name.
  - URL's in the spreadsheet are now treated as active hyperlinks.
  - Row heights can be adjusted all at once, to interactively scale (and hide) molecular depictions.
  - The VIDA Name column has been removed.
  - New columns can be added via a "+" button in the spreadsheet headers
- Spreadsheets can now be exported directly to PDF files.
- Filenames of files opened within **VIDA** now show in the title bar of **VIDA**'s main window.
- The behavior of the "Recents" menu has been changed, so that it now displays both recently opened and recently saved files, including **VIDA** state (.oes) files. In addition, a "Recent Directories" menu has been added.
- A "protein sequence" coloring scheme has been added which colors protein molecules using a gradient from N-terminus (blue) to C-terminus (red).
- The Extension Manager has been rewritten to simplify the user interface.
- **VIDA** can now automatically subscribe to sets of VIDA Extensions pointed to by the environment variable, VIDA_EXTENSIONS_URL.

- Several rarely-used icons have been removed from **VIDA**'s main window, and the "Style Toolbar" at the top of the 3D window is now empty by default.
- An example docking extension is now provided, to allow docking of molecules against a pre-prepared receptor for customers with an **OEDocking TK** license.
- New commands for disabling and clearing **VIDA**'s "Undo" stack have been added to the scripting API. These can be used to prevent repeated operations from building up large amounts of "undo" memory unnecessarily.
- **VIDA** can now read MTZ crystallographic reflection files from several common refinement programs and generate regular and difference maps from them.
- A new preference setting allows molecules without 3D coordinates to be displayed in the 3D window.
- Monitors (for distances, torsions, etc.) can now be deleted from within the "Measure" option in the Mouse Toolbar.
- **VIDA** now incorporates the released versions of OpenEye toolkits, including the newer **OEDocking TK**, **OEDepict TK**, and **Grapheme TK**.
- The Preferences dialog behavior has been modified to allow updating of preferences without dismissing the dialog.
- The installer for Windows now allows association of user-selectable file types with **VIDA**.
- Load times have been significantly improved for files containing large numbers of molecules.

18.11.2 Major Bug Fixes

- Under certain conditions, working with a protein with alternate conformations could cause corruption of state files. This has been corrected.
- A problem causing failure of molecule check-ins after deleting and re-adding hydrogens has been fixed.
- A severe graphics slowdown on Windows machines with Intel HD Graphics video has been addressed by new OpenGL rendering methods.
- A bug which caused atom names to sometimes be lost when merging molecules has been fixed.
- Stereochemistry is now preserved when pasting structures from ChemDraw or reading .cdx files.
- A graphics bug which made all 3D labels show up in black on certain video cards has been corrected.

18.11.3 Minor Bug Fixes

- Molecules created by typing a name in the builder would be named with a SMILES string instead of the type-in name, unless a valid **Lexichem TK** license was found.
- MoleculeColorByScope() and related functions have been fixed to work properly with molecule keys in the Scratch Scope.
- Molecule surfaces are now automatically updated when the molecule is edited.
- On machines with desktops spanning multiple monitors, **VIDA** could sometimes place the main window and Preferences dialog offscreen.
- The **VIDA** icon now shows correctly when the application is pinned to the taskbar in Windows.
- Disabled inner contours from **FRED** receptor files are no longer displayed.
- Previously, users could see very slow performance in the File/Save dialog, when trying to select all molecules with the topmost checkbox. This problem has been eliminated.

18.11.4 Platform support

Support for Ubuntu 12 (x64) and Mac OS X 10.7 has been added.

18.12 VIDA 4.1.1

November 2011

18.12.1 New Features

- Ability to save and restore preferences in extensions. Many of the example extensions were updated to make use of this new capability, and now preserve settings between **VIDA** sessions.
- Capability to customize molecule lookup and property calculation functions in the builder.
- Adjustable jitter and a new coloring method for grids rendered using the "cloud" style.
- Ability to disable OpenGL hardware rendering via Preferences, to circumvent issues with certain video drivers (most notably, the Intel HD Graphics drivers for Windows).

18.12.2 Major Bug Fixes

- Fixed a crash that sometimes occurred when performing distance selection on proteins.
- Addressed an issue which could cause bond visibilities to become corrupted.
- Fixed an issue which could cause long delays entering build mode in sessions with large molecules.
- Multi-conformer molecules with child molecules were not being displayed properly in the List Window. This has been fixed.
- Resolved a crash that could occur during torsion measurements on Linux machines.
- Fixed a problem with excessive memory consumption when surfaces were scribed and cropped multiple times.
- Corrected a severe slowdown caused by an interaction of some extensions with a molecule selection dialog.
- Fixed an issue which made check-ins of multiconformer molecules very slow.

18.12.3 Minor Bug Fixes

- Changed the way FRED View displays binding site residues in the absence of a reference ligand.
- Lasso selection in the 2D Sketcher was leaving hydrogens in the 3D window unselected. This has been corrected.
- An issue preventing **VIDA** from reading files with capitalized or mixed-case file extensions was fixed.
- A problem which caused torsion measurements on Windows platforms to be incorrectly rendered was fixed.
- Fixed a bug which prevented users from being able to set surface coloring defaults.
- Made performance improvements to some of the **VIDA** extensions.

18.13 VIDA 4.1.0

August 2011

18.13.1 New Features

- Simultaneous 2D and 3D editing of small molecules.
- Ability to rotate and translate molecules independently in the builder.
- Simplified authoring of extensions, and inclusion of a number of new example extension scripts.
- Ability to add annotations to existing objects, and have those annotations appear in the List Window.
- Adjustment of torsion angles within rings to change ring puckers and interconvert axial and equatorial substitutions.
- Option to use the MMFF94s force field when building molecules.
- Improved set of fragment templates in the sketcher

18.13.2 Major Bug Fixes

- Fixed a crash when saving state files after loading a large number of proteins.
- Fixed a crash when editing cells in a spreadsheet.
- Centering on selected atoms would sometimes require a change in visibility before taking effect. This has been fixed.
- Fixed a bug which caused rows to sometimes disappear when cells in a spreadsheet were edited.
- Spreadsheet depictions are now being updated as molecules are edited.
- An issue which caused the view to occasionally jump unexpectedly during molecule rotations was fixed.
- Fixed a problem which sometimes caused crashes while creating measurements in the “All” scope.
- Changing colors/styles of grids now operates on the proper scope, rather than on all visible grids.
- Addressed issues with the builder and sketcher which sometimes caused fragments to be joined incorrectly.
- Fixed a bug which sometimes caused right-click menus to become inoperative.

18.13.3 Minor Bug Fixes

- Hotkeys for Marked/Locked/Visible now work on selected rows in the spreadsheet, not just the active molecule.
- Fixed a bug where changes to the spreadsheet cell formats weren’t being propagated immediately to the spreadsheet.
- Fixed a bug to have **VIDA** remember the hidden/visible state of depictions in a spreadsheet from one session to another.
- Fixed an issue with the spreadsheet so that users can’t edit non-editable cells, such as computed values (e.g., molecular weight, number of rotors) and user-defined python expressions.
- Fixed a sizing problem with the 2D Sketcher window, which had caused some tools in its toolbar to be hidden.
- When a spreadsheet is created from a list, it is now made the current spreadsheet.
- Fixed an issue which would cause blank entries to sometimes appear in the List Window.
- Fixed a rendering problem which caused circles to be displayed incorrectly.
- Corrected an issue in which text would be incorrectly wrapped within labels.
- Display of symmetry-related molecules now updates automatically as the view is translated.
- Surface created in the FRED View are now added to the List Window and can be deleted individually.
- Grid contours are now contoured by default at the original resolution of the grid file.

- Contours of a grid can now be temporarily shown, similarly to conformers, by clicking on their names in the List Window.
- All child objects of grids now show in the List Window. Previously only the contours would show.
- Corrected an issue which sometimes caused incorrect hydrogen geometries during building.
- Addressed several issues affecting performance and behavior of spreadsheets.
- Fixed an issue that could sometimes cause incorrect stereochemistries to appear in 2D depictions.
- Fixed a bug which caused stereochemistry to be omitted from molecules when copying to the clipboard.
- Corrected strange lasso behavior in the sketcher.

18.13.4 Platform support

Support for Ubuntu 10 (x64) and RedHat Enterprise v6 (x64) was added. RedHat Enterprise v3 is no longer supported.

18.14 VIDA 4.0.3

June 2010

18.14.1 Improvements

- Fixed extension loading to enforce rules more consistently & clearly. Added a dialog to notify a user when an extension fails to load at startup.
- Menu items which throw exceptions other than a cancel runtime exception now will display a dialog rather than failing quietly.
- Bookmark widget buttons are a touch larger to make them easier to click.
- Augmented menu API to add functions MenuGetAllItems, MenuItemIsAMenu, MenuDisplayNameGet and MenuDisplayNameSet. It should now be possible to determine complete menu structure at runtime.
- Sketching a new molecule is much improved. It now features a simple click and click-and-drag interface.
- The invert stereo button in the 2D sketcher & 3D editor is now the default. It also has a slightly improved icon.

18.14.2 Major Bug Fixes

- Hitting the down arrow in the spreadsheet could result in the current row being reset to the first row, making it impossible to scroll with the arrow keys.
- After a lot of editing & deletion, the list window could end up with a corrupted display.
- Some children objects were not properly preserving visibility in state files. Affected types included monitors, and surfaces and grids which were toggled in the style control.
- Files containing Unicode characters previously might fail to load. They should now load properly.
- Improperly constructed extensions could cause problems for the extension manager. The previously might look loaded even though they were not. **VIDA** now handles erroneous extensions correctly.
- The right click “Minimize” option in builder mode attempted to call a non-existent function
- Screenshots in .jpg format resulted in empty files on Microsoft Windows
- In some cases, the save dialog would only allow a single item to be clicked.

18.14.3 Minor Bug Fixes

- Stereochemistry in the 2D depictions was not displayed in the spreadsheet or the 2D preview window.
- On some platforms, the graphics widget that controls slabbing and depthcue in the style control can dramatically slow down VIDA, especially for spreadsheet operations. When it is collapsed, this slow down does not occur. As a temporary work around, the collapsed state of the style control (including the graphics widget) is now remembered between sessions.
- Documented the AppVersion, AppVersionMajor, AppVersionMinor and AppVersionBugFix commands.

18.15 VIDA 4.0.0

AuguAgust 2009

18.15.1 New Features

- Added a 1D/2D molecule input dialog which supports SMILES, names, and interactive building.
- Added a 3D editing mode which enables significant modification of the current molecule including in-vacuo minimization with MMFF94.
- Added support for DNA/RNA ribbon style visualization
- Added support for displaying and controlling alternate residue conformations (turn on *ALL* flavor when reading PDB files to enable this) as well as a number of coloring schemes associated with alternate residues
- 2D depiction in 3D display is now selectable
- Add object-type specific tool buttons to the bottom display which provide context sensitive operations such as coloring and style options.
- Added the ability to import all of OpenEye's Python toolkits for use within VIDA, provided that the user has a license for those toolkits.
- Added the ability to transfer molecules, grids, and surfaces between the main application and the bundled OpenEye Python toolkits.
- Added support for PythonQt scripting within **VIDA**.
- Added support for PyQt scripting with **VIDA** (separate license required from Riverbank computing).
- Added interactive protein depictions in the spreadsheet.
- Added many new atom properties to the atom spreadsheet.
- Added a residue spreadsheet which is capable of controlling alternate residue conformations.
- Added the ability to load molecules directly from the PDB ("File > Open Special > From PDB")
- Added the following command line arguments:

  `-e` - Runs the associated file through the Python interpreter and populates the sys.argv variable with the remaining command line arguments

  `-eonview` - Loads the associated files and processes them using the built in *EON* view

  `-fredview` - Loads the associated files and processes them using the built in *FRED* view

  `-rocsview` - Loads the associated files and processes them using the built in *ROCS* view

  `-import` - Loads the associated files and immediately launches the spreadsheet import dialog
- **PICTO** application now ships with **VIDA**. **PICTO** requires a valid **VIDA** license.

18.15.2 Improvements

- Significantly increased the speed of searches performed within **VIDA**.
- Significantly improved and expanded the scripting examples that come with the distribution.
- Significantly optimized the loading of OEB molecule files which contain attached grids.
- When displaying *Marked* molecules, now only the carbon atoms (as opposed to all) are colored using the *Marked* color unless the non-carbon atoms are specifically *Marked* themselves.
- Optimized the AtomDataGet function when working with multi-conformer molecules.
- Vastly improved the display and usability of the embedded 3D toolbars

18.15.3 Major Bug Fixes

- Fixed rendering problems associated with having multiple overlapping transparent surfaces displayed simultaneously.
- Fixed the handling of triple-quoted strings in the **VIDA** interpreter.
- Fixed a bug which was preventing scripts from being aborted while running.
- Fixed a problem where SD data was being lost on read if one or more conformers in a multiconformer molecule did not have the same SD data fields as the rest.

18.15.4 Minor Bug Fixes

- Fixed a number of erroneous tooltips.
- Fixed a few spelling errors in prompts and widgets.
- Fixed the behavior of the drop-down selections in the constraints and extra molecule buttons in the FRED View controller.
- Fixed the hybridization, isotope, and symmetry atom labels.
- Fixed a bug where the ColorSetScoped function did not color selected atoms if SelectedScope was explicitly specified as opposed to BestScope.
- Fixed the MoleculeMergeScoped function to ensure that residue information is not lost when two molecules are merged together.
- Fixed the standard deviation function in the spreadsheet.
- The "Create Spreadsheet" option for list entries in the list windows now works as expected.
- Fixed the coordinates returned in the AtomData class by the AtomDataGet function when working with multi-conformer molecules.
- Silenced a large number of annoying and unnecessary warnings that would appear in the scripting window

18.16 VIDA 3.0.0

August 2011

18.16.1 New Features

- Added dynamic help system
- Added new 2D depiction color schemes: black on white, white on black, color on white and color on black

- Added new molecule coloring schemes: amino, chain, group, cpk, cpknew, shapely, bfactor, formal charge, and partial charge
- Added new surface hydrophobicity scales: charifson, eisenberg, kyte-dolittle, and white-octanol
- Added support for reading attached grids and surfaces in OEB files
- Added support for reading **Spicoli** geometric annotations
- Added support for reading **FRED** receptor files
- Added new hydrogen style: polar hydrogens only
- Added multiple new atom and bond label types (including one for **OEChem** generic data)
- Added new 3D display mouse maps for Coot and PyMol

18.16.2 Improvements

- Anti-aliasing of text and 2D depictions
- Much better control over placement of dock windows
- Dock windows can be stacked on top of each other in tabs
- Single toolbar at the top of the application
- Additional popup toolbars placed around edge of main window
- Enabled grid contour transparency
- Hydrogen style is now a molecule specific property instead of a global property
- C-Alpha trace display is now a molecule specific property instead of a global property
- Ribbon display is now a molecule specific property instead of a global property
- Many miscellaneous 3D rendering improvements and accelerations
- Many miscellaneous user interface improvements

18.16.3 Major Bug Fixes

- Upgraded to OEChem 1.5 which fixed a number of bugs

18.16.4 Minor Bug Fixes

- Fixed a problem on some graphics cards where only aromatic rings would be drawn when a molecule was in stick mode

18.17 VIDA 2.1.2

18.17.1 New Features

- Added “Copy Image” function under the Edit menu which takes a screenshot of the current main window and places it on the clipboard
- Added “Map Index” atom label corresponding to the value returned by the OEChem function OEAtomBase::GetMapIdx().

18.17.2 Improvements

- Updated menu system to be compliant with standard UI design practices (such as '…' usage when menu option launches a dialog)
- Updated menu system on Mac OS X to follow common Mac paradigms (e.g. "Open Recents" and "Preferences…" options)
- Reorganized File menu for more logical placement of options
- Collapsed "New > List >" submenu structure to a single "New List >" submenu which appears after the Open, Save block
- Added "New Molecules >" submenu which contains a "From Split" submenu which provides the "Split Molecules" functionality from the old Edit menu.
- Renamed "Clear All" option to "Clear" and moved up in the menu above Save block
- Removed "Abort Script" option as it was redundant (Stop button next to progress bar performs this function) and also could generally not be accessed while a script was running
- Reorganized Edit menu for more logical placement of options as well as to provide some commonly performed operations in a single place
- Moved "Split Molecules" functionality to File menu
- Added new "Select" submenu which contains the following options: All Visible, Query, Invert, and None
- Added new "Mark" submenu which contains the following options: All, Query, None
- Added new "Hide" submenu which contains the following options: All, Selected, Marked, Not-Selected, Not-Marked, and None
- Removed the following options as they were redundant given the above additions: Hide All, Clear Marked
- Alphabetized list of available windows in the Window menu
- Batch shell window no longer appears when running on Windows
- Redesigned process manager window for easier use
- Significantly improved the quality of POV-ray exports and reduced the output file size
- Improved the rendering of electrostatic grids

18.17.3 Major Bug Fixes

- Upgraded to OEChem 1.4.1 which fixed a number of bugs associated with handling bad input files
- Fixed a cross-platform compatibility issue in state files

18.17.4 Minor Bug Fixes

- Fixed "MolecularEnergy()" calculation method in spreadsheet to correctly report energies on multi-conformer molecules
- Fixed "SpreadsheetFilter()" command such that it operates on the specified spreadsheet instead of the current spreadsheet
- Ensured that all current environment variables are properly passed to processes started from the process manager
- Fixed the icon on the hydrogen display button to accurately reflect what will occur when the button is pressed
- Fixed a Mac OS X bug where double-clicking on a molecule file started **VIDA** but did not actually load the file that was clicked on

18.18 VIDA 2.1.1

18.18.1 Improvements

- Filled out Sybyl mouse map in more detail
- Added "fixed size text" option for labels (see Preferences) such that the label font size remains fixed independent of zooming in the scene, however the font size can be adjusted using the "Text Scale" slider in the Style Bar window
- Added the ability to specify the font for labels when using "nice fonts" for labels (see Preferences)
- Added a "custom" option to the atom and bond label dialogs to allow for easier specification of custom labels without having to use scripting commands
- Improved the drawing of labels such that they always appear in front of the object they are labeling
- Improved the ability to see the selected state of partially selected residues in the sequence view
- Added file dialog filter option to allow reading of ASCII formatted grids (*.agd*)
- Added support for native builds on Intel Macs

18.18.2 Major Bug Fixes

- Fixed a problem with reading spreadsheet information in state files generated by **VIDA** 2.0.x which caused extra tabs to appear in the spreadsheet
- Fixed a problem with writing spreadsheet information correctly in **VIDA** 2.1
- Fixed a crash when reading a badly formatted state file (potentially generated by the bug fixed above)
- Fixed a crash when attempting to import a nonstandard whitespace delimited spreadsheet (whitespace is not a supported spreadsheet delimiter)
- Fixed a crash on certain video cards when attempting to display "fixed size text" labels
- Fixed a crash caused by dragging state files onto the application icon in order to open them
- Fixed an application hang when reading badly formatted XPLOR maps
- Fixed an application hang when reading a badly formatted OEB file
- Fixed a memory leak generated when running a large number of scripting commands

18.18.3 Minor Bug Fixes

- Fixed a problem where extra atom labels could appear on molecules when saved to state
- Fixed a problem where CTRL-mouse over atom displays were not seen when using non-VIDA mouse maps
- Fixed a bug in the "Text Scale" slider where its position did not always reflect the current scale being used
- Fixed a bug where `OE_DIR` and `OE_LICENSE` environment variables were not being properly exported to the Process Manager on Mac OS X such that spawned applications could not find a valid `oe_license.txt` file
- Fixed a bug where exporting a CSV file on Microsoft Windows attached an extra return character causing a blank line to appear between each row when read into Microsoft Excel
- Fixed a few cases where the display of depictions in the spreadsheet was not obeying the set preferences
- Fixed a bug where extra calls to undo cleared the redo stack
- Turned off the writing of empty data fields when writing molecules to SD or OEB formats. This can be turned back on if truly desired from the Preferences

18.19 VIDA 2.1.0

18.19.1 New Features

- Added interactive text annotation of molecules
- Added the ability to start, monitor, terminate, and load results from external processes
- Added a protein sequence viewer as part of the 2D depiction widget
- Added the `oechemlite` Python module, enabling significantly greater scripting functionality within VIDA
- Added the ability to create arbitrary sphere monitors via the scripting API (see Python API for documentation of the new method `CreateSphereMonitor(...)`)
- Added support for fullscreen display
- Added new emph{cloud} visualization style for grids
- Added support for copy and paste of molecules from ISIS/Draw
- Added support for use of the Maestro mouse map in the 3D display
- Added support for saving screenshots in JPG format in addition to PNG

18.19.2 Improvements

- Cleaned up the appearance of the right-click popup menus
- Added a "Style" option to the right-click popup menus, when appropriate to the context, which mirrors the main "Style" menu
- Added significantly greater detail to info labels created when holding down the "ctrl" (or "alt" on Mac) key when moving the mouse in the 3D or 2D views
- Added two default filtered spreadsheets to the application: "Molecules" and "Proteins"
- Filled out the scripting API for greater control of the GUI - in particular for menus and toolbar
- Greatly improved the electrostatic coloring scheme for protein surfaces
- Added the ability to change the font used by the application

18.19.3 Major Bug Fixes

- Fixed a crash when trying to generate electrostatic contours on extremely small molecules (e.g. water)

18.19.4 Minor Bug Fixes

- Fixed a number of preferences that were not being properly saved or obeyed by the application
- Fixed the undo stack such the initial setup of the 3D display cannot be undone by accident
- Fixed the display of angle monitor measurements from radians to degrees
- Fixed the display of the text scale widget such that it is not hidden on low resolution monitors
- Made sure that when saving images and state files, a proper file extension is added to the filename, even if one was not specified

18.20 VIDA 2.0.2

18.20.1 Improvements

- Added "Style" menu to the main menubar which replicates the functionality of the style control (useful in low-resolution settings)
- Added "Center" and "Color" options to the popup dialogs
- Enabled display of named network drives in the file dialog
- Enabled use of native file dialogs instead of custom **VIDA** file dialog as a preference.
- Improved 3D visualization support for lower-end graphics cards
- Increased default line width in 3D scene for better visibility
- Enabled transparent display of grids (via the "Style" menu)
- Added simple spreadsheet import preview dialog to help select the proper column delimiter (tab vs. comma)
- Added `SpreadsheetCurrentSet()` function which allows specification of current spreadsheet via scripting interface
- Improved the SD file reader for better handling of badly formatted SD files
- Improved the handling of situations in which the atom indices specified in input files differ from the order in which they are specified
- Improved start-up time as license detection mechanisms have been significantly improved

18.20.2 Major Bug Fixes

- Fixed right-click on empty 2D depiction crash
- Fixed spreadsheet import failure on linux
- Fixed mainwindow snapshots such that file dialog does not obscure the image to be captured
- Fixed license detection routines such **VIDA** will properly detect installed licenses and not cause start-up problems if the license file is moved
- Fixed all other reported crashes not otherwise listed

18.20.3 Minor Bug Fixes

- Fixed a bug in the detection of a user's home directory in networked environments on Microsoft Windows
- Fixed a bug in "measurement" mode where the first atom or bond picked was not being drawn as selected
- Fixed a bug where spreadsheet filtering performed via the scripting interface would not behave as expected if the specified spreadsheet was not the same as the current spreadsheet
- Fixed an uncommon situation in which the locked and visible controls in the list window do not work as expected
- Fixed the handling of certain "escape characters" in filenames specified on the commandline, files dragged onto the **VIDA** icon, and the "startup.py" file such that they no longer prevent the opening of those files

CHAPTER

NINETEEN

CITATION

Note

To cite VIDA please use the following:

VIDA <vida_version>. OpenEye, Cadence Molecular Sciences, Santa Fe, NM. http://www.eyesopen.com

The version number is available on the front page of the VIDA section.

CHAPTER

TWENTY

COPYRIGHT AND TRADEMARKS

© 1997-2024 Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA. All rights reserved.

OpenEye, Lexichem, ROCS, Grapheme, and Orion are registered trademarks of Cadence Design Systems, Inc.

This material contains proprietary information of Cadence Design Systems, Inc. Use of copyright notice is precautionary only and does not imply publication or disclosure.

The information supplied in this document is believed to be true, but no liability is assumed for its use or the infringement of the rights of others resulting from its use. Information in this document is subject to change without notice and does not represent a commitment on the part of Cadence Design Systems, Inc.

This package is sold/licensed/distributed subject to the condition that it shall not, by way of trade or otherwise, be lent, re-sold, hired out, or otherwise circulated without Cadence Design Systems' prior consent, in any form of packaging or cover other than that in which it was produced. No part of this manual or accompanying documentation, may be reproduced, stored in a retrieval system on optical or magnetic disk, tape, CD, DVD or other medium, or transmitted in any form or by any means, electronic, mechanical, photocopying recording, or otherwise for any purpose other than for the purchaser's personal use without a legal agreement or other written permission granted by Cadence.

This product should not be used in the planning, construction, maintenance, operation or use of any nuclear facility nor the flight, navigation, or communication of aircraft or ground support equipment. Cadence Design Systems, Inc., shall not be liable, in whole or in part, for any claims arising from such use, including death, bankruptcy or outbreak of war.

Windows is a registered trademark of Microsoft Corporation. Apple, OS X, macOS, and Macintosh are registered trademarks of Apple Inc. UNIX is a registered trademark of the Open Group. Linux is a registered trademark of Linus Torvalds. Red Hat is a registered trademark of Red Hat, Inc. SUSE, SLED, and SLES are registered trademarks of Novell, Inc. Ubuntu is a registered trademark of Canonical Ltd.

SYBYL is a registered trademark of Certara, L.P. MDL, BIOVIA and ISIS are registered trademarks of Dassault Systemes and/or its affiliates. SMARTS and SMIRKS are registered trademarks of Daylight Chemical Information Systems, Inc. MacroModel is a registered trademark of Schrodinger, LLC.

Python is a registered trademark of the Python Software Foundation. Django is a registered trademark of the Django Software Foundation. Java is a registered trademark of Oracle and/or its affiliates.

Other products and software packages referenced in this document are trademarks and registered trademarks of their respective vendors or manufacturers.

CHAPTER

TWENTYONE

SAMPLE CODE

For all of the code examples and samples within documents published by OpenEye, Cadence Molecular Sciences, the following terms of use apply.

TERMS FOR USE OF SAMPLE CODE

The software below ("Sample Code") is provided to current licensees or subscribers of Cadence products or SaaS offerings (each a "Customer"). Customer is hereby permitted to use, copy, and modify the Sample Code, subject to these terms. Cadence claims no rights to Customer's modifications. Modification of Sample Code is at Customer's sole and exclusive risk. Sample Code may require Customer to have a then current license or subscription to the applicable Cadence offering.

THE SAMPLE CODE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. CADENCE DISCLAIMS ALL WARRANTIES, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. In no event shall Cadence be liable for any damages or liability in connection with the Sample Code or its use.

CHAPTER

TWENTYTWO

CITATION

Please follow the guidelines in this document to cite use of our products in research publications and other documents.

For more information, please contact OpenEye Support.

22.1 Orion®

To cite use of Orion, please use the following:

```
Orion <version-number>. OpenEye, Cadence Molecular Sciences, Santa Fe, NM. http://www.
↪eyesopen.com.
```

For example:

```
Orion 2025.1. OpenEye, Cadence Molecular Sciences, Santa Fe, NM. http://www.eyesopen.com.
```

The version number for Orion appears in the product's release notes, for example, https://docs.eyesopen.com/orion-user-interface/releasenotes/index.html.

22.2 Orion Floes

To cite use of an Orion-based floe package, please use the following:

```
OpenEye <package-name> <version-number>. OpenEye, Cadence Molecular Sciences, Santa Fe,␣
↪NM. http://www.eyesopen.com.
```

For example:

```
OpenEye Large-Scale Floes 4.2.3. OpenEye, Cadence Molecular Sciences, Santa Fe, NM.␣
↪http://www.eyesopen.com.
```

The version number for a floe package is displayed on the first page of the package's release notes. For example: https://docs.eyesopen.com/floe/modules/large_scale_floes/docs/source/releasenotes_2025_1.html.

22.3 Toolkits and Applications

To cite OpenEye toolkits or applications used in your work, please use the following:

```
OpenEye Toolkits [or Applications] <version-number>. OpenEye, Cadence Molecular Sciences,
↪ Santa Fe, NM. http://www.eyesopen.com.
```

For example:

```
OpenEye Toolkits 2024.2.1. OpenEye, Cadence Molecular Sciences, Santa Fe, NM. http://www.
↪eyesopen.com.
```

The Applications and Toolkits are released as a package, multiple times per year. The first part of the version number designates the calendar year. The version number appears on the summary document for the package.

If you want to cite an individual application or toolkit, you can use the syntax:

```
<product name> <version-number>. OpenEye, Cadence Molecular Sciences, Santa Fe, NM.␣
↪http://www.eyesopen.com.
```

where <*product name*> is:

- Bioisostere TK
- OEChem TK
- OEDepict TK
- OEDocking TK
- OEFF TK
- Eon TK
- FastROCS TK
- Grapheme TK
- GraphSim TK
- Lexichem TK
- OEMedChem TK
- MolProp TK
- Omega TK
- Quacpac TK
- Shape TK
- Spicoli TK
- Spruce TK
- Szmap TK
- Szybki TK
- Zap TK
- AFITT Application
- BROOD Application
- OEDocking Application
- EON Application
- OMEGA Application
- PICTO Application
- pKa-Prospector Application
- QUACPAC Application

- ROCS Application
- SiteHopper Application
- SPRUCE Application
- SZMAP Application
- SZYBKI Application
- VIDA Application

You can use the version number of the individual Application or Toolkit. For example:

```
SZYBKI Application 2.8.0.1. OpenEye, Cadence Molecular Sciences, Santa Fe, NM. http://
↪www.eyesopen.com.

Szybki TK 2.8.0.1. OpenEye, Cadence Molecular Sciences, Santa Fe, NM. http://www.
↪eyesopen.com.
```

The first documentation pages give the product names and version numbers, as in https://docs.eyesopen.com/applications/szybki/index.html and https://docs.eyesopen.com/toolkits/python/szybkitk/index.html.

22.4 Publications for Bibliographies

Please also cite these publications if the corresponding product was used in your research.

22.4.1 Orion

Sørensen, J., Bannan, C.C., Calabrò, G., Jain, V., Ovanesyan, G., Smith, A., Zhang, S., Bayly, C.I., Darden, T.A., Geballe, M.T., LeBard, D.N., McGann, M., Moon, J.B., Muddana, H.S., Shewmaker, A., LaFon, J., Tolbert, R.W., Skillman, A.G., Nicholls, A. Orion® A Cloud-Native Molecular Design Platform. In *Computational Drug Discovery*. V. Poongavanam and V. Ramaswamy, Eds. Wiley-VCH, 2024; pp. 579–615. DOI: 10.1002/9783527840748.ch24

22.4.2 AFITT and FLYNN Applications

Wlodek, S.; Skillman, A.G.; Nicholls, A. Automated Ligand Placement and Refinement with a Combined Force Field and Shape Potential. *Acta Crystallogr., Sect. D: Struct. Biol.* **2006**, *62* (7), 741–749. DOI: 10.1107/s0907444906016076

22.4.3 OEDocking Application and Toolkit

Kelley, B.P.; Brown, S.P.; Warren, G.L.; Muchmore, S.W. POSIT: Flexible Shape-Guided Docking For Pose Prediction. *J. Chem. Inf. Model.* **2015**, *55* (8), 1771-1780. DOI: 10.1021/acs.jcim.5b00142

McGann, M. FRED Pose Prediction and Virtual Screening Accuracy. *J. Chem. Inf. Model.* **2011**, *51* (3), 578-596. DOI: 10.1021/ci100436p

McGann, M. FRED and HYBRID Docking Performance on Standardized Datasets. *J. Comput.-Aided Mol. Des.* **2012**, *26* (8), 897-906. DOI: 10.1007/s10822-012-9584-8

22.4.4 OMEGA Application and Toolkit

If you are using OMEGA in torsion driving mode, please cite the following two publications.

Hawkins, P.C.D.; Skillman, A.G.; Warren, G.L.; Ellingson, B.A.; Stahl, M.T. Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and the Cambridge Structural Database. *J. Chem. Inf. Model.* **2010**, *50* (4), 572-584. DOI: 10.1021/ci100031x

Hawkins, P.C.D. and Nicholls, A. Conformer Generation with OMEGA: Learning from the Data Set and the Analysis of Failures. *J. Chem. Inf. Model.* **2012**, *52* (11), 2919–2936. DOI: /10.1021/ci300314k

If you are using OMEGA in macrocyle mode, please cite the following.

Hawkins, P.C.D. and Wlodek, S. Decisions with Confidence: Application to the Conformation Sampling of Molecules in the Solid State. *J. Chem. Inf. Model.* **2020**, *60* (7), 3518–3533. DOI: /10.1021/acs.jcim.0c00358

22.4.5 ROCS Application

Hawkins, P.C.D.; Skillman, A.G.; Nicholls, A. Comparison of Shape-Matching and Docking as Virtual Screening Tools. *J. Med. Chem.* **2007**, *50* (1), 74-82. DOI: 10.1021/jm0603365

22.5 OpenEye MMDS Web Service

To cite use of the Macromolecular Data Service (MMDS) web service, please use the syntax:

```
Macromolecular Data Service <version-number>. OpenEye, Cadence Molecular Sciences, Santa␣
↪Fe, NM. http://www.eyesopen.com.
```

For example:

```
Macromolecular Data Service 1.1. OpenEye, Cadence Molecular Sciences, Santa Fe, NM.␣
↪http://www.eyesopen.com.
```

The MMDS version number appears on the web service's release notes.

CHAPTER

TWENTYTHREE

TECHNOLOGY LICENSING

OpenEye products use the following technology under license.

Some of the open source technologies we use are licensed under the GNU Public License (GPL) version 2 or 3. In all instances, these technologies are communicated with using the "at arms length" principle, using pipes and command line arguments.

While the software is in some instances assembled into a container, that assembly does not invoke the GPL copyleft scope. For each Floe package where these are used, the environment log clearly details the installed programs and their version numbers. With the links below for each technology, the source for each of those programs is available.

Table 1: Licensed Technology

Name of Project	Website	License URL
_libgcc_mutex	https://github.com/conda-forge/_libgcc_mutex-feedstock	https://github.com/conda-forge/_libgcc_mutex-feedstock/blob/master/LICENSE.txt
_openmp_mutex	https://github.com/conda-forge/_openmp_mutex-feedstock	https://github.com/conda-forge/_openmp_mutex-feedstock/blob/main/LICENSE.txt
ablang2	https://github.com/TobiasHeOl/AbLang2	https://github.com/TobiasHeOl/AbLang2/blob/main/LICENSE
abseil-cpp	https://github.com/abseil/abseil-cpp	https://github.com/abseil/abseil-cpp/blob/master/LICENSE
absl-py	https://github.com/abseil/abseil-py	https://github.com/abseil/abseil-py/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
aiohappyeyeballs	https://github.com/aio-libs/aiohappyeyeballs	https://github.com/aio-libs/aiohappyeyeballs/blob/main/LICENSE
aiohttp	https://docs.aiohttp.org/en/stable/	https://github.com/aio-libs/aiohttp/blob/master/LICENSE.txt
aiosignal	https://github.com/aio-libs/aiosignal	https://github.com/aio-libs/aiosignal/blob/master/LICENSE
Amazon Linux 2	https://aws.amazon.com/amazon-linux-2	N/A
AmberUtils	http://ambermd.org	N/A
ambit	https://github.com/khimaros/ambit	https://github.com/khimaros/ambit/blob/master/LICENSE
amqp	https://github.com/celery/py-amqp	https://github.com/celery/py-amqp/blob/master/LICENSE
anaconda-anon-usage	https://github.com/anaconda/anaconda-anon-usage	https://github.com/anaconda/anaconda-anon-usage/blob/main/LICENSE
angular	https://github.com/angular/angular.js	https://github.com/angular/angular.js/raw/master/LICENSE.md
angular-animate	https://github.com/angular/angular.js	https://github.com/angular/angular.js/raw/master/LICENSE.md
angular-cache	https://github.com/jmdobry/angular-cache	https://github.com/jmdobry/angular-cache/raw/master/LICENSE
angular-cookies	https://github.com/angular/angular.js	https://github.com/angular/angular.js/raw/master/LICENSE.md

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
angular-loggly-logger	https://github.com/ajbrown/angular-loggly-logger	https://github.com/ajbrown/angular-loggly-logger raw/master/LICENSE.md
angular-mocks	https://github.com/angular/angular.js	https://raw.githubusercontent.com/angular/angular.js/master/LICENSE
angular-resource	https://github.com/angular/angular.js	https://raw.githubusercontent.com/angular/angular.js/master/LICENSE
angular-toggle-switch	https://github.com/cgarvis/angular-toggle-switch	https://github.com/cgarvis/angular-toggle-switch blob/master/LICENSE
angular-ui-grid	https://github.com/angular-ui/ui-grid	https://github.com/angular-ui/ui-grid
angular-ui-router	https://github.com/angular-ui/ui-router	https://github.com/angular-ui/ui-router/raw/master/LICENSE
angular-ui-tree	https://github.com/angular-ui-tree/angular-ui-tree	https://github.com/angular-ui-tree/angular-ui-tree/raw/master/LICENSE
angular-vs-repeat	https://github.com/kamilkp/angular-vs-repeat	https://raw.githubusercontent.com/kamilkp/angular-vs-repeat/master/LICENSE
aniso8601	https://bitbucket.org/nielsenb/aniso8601	https://bitbucket.org/nielsenb/aniso8601/src/master/LICENSE
annotated-types	https://github.com/annotated-types/annotated-types	https://github.com/annotated-types/annotated-types/blob/main/LICENSE
antlr4-python3-runtime	http://www.antlr.org OR https://pypi.org/project/antlr4-python3-runtin	https://github.com/conda-forge/antlr4-python3-runtin blob/master/LICENSE.txt

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
anyio	https://github.com/agronholm/anyio	https://github.com/agronholm/anyio/blob/master/LICENSE
appdirs	http://github.com/ActiveState/appdirs	http://github.com/ActiveState/appdirs/blob/master/LICENSE.txt
appengine	https://google.golang.org/appengine	https://github.com/golang/appengine/blob/master/LICENSE
arabic-reshaper	https://github.com/mpcabd/python-arabic-reshape	https://github.com/mpcabd/python-arabic-reshape/blob/master/LICENSE
archspec	https://github.com/archspec/archspec/blob/master/README.md	https://github.com/archspec/archspec/blob/master/COPYRIGHT
argon2-cffi	https://github.com/hynek/argon2-cffi	https://github.com/hynek/argon2-cffi/blob/master/LICENSE
argon2-cffi-bindings	https://github.com/hynek/argon2-cffi-bindings	https://github.com/hynek/argon2-cffi-bindings/blob/master/LICENSE
arpack	https://www.caam.rice.edu/software/ARPACK/	https://www.caam.rice.edu/software/ARPACK/
arrow	https://github.com/arrow-py/arrow	https://github.com/arrow-py/arrow?tab=Apache-2.0-1-ov-file#readme
ascii85	https://github.com/huandu/node-ascii85	https://github.com/huandu/node-ascii85/raw/master/LICENSE
ase	https://wiki.fysik.dtu.dk/ase/	https://wiki.fysik.dtu.dk/ase/development/licenseinfo.html#license-info
asgiref	https://github.com/django/asgiref/	https://github.com/django/asgiref/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
asn1crypto	https://github.com/wbond/asn1crypto	https://github.com/wbond/asn1crypto/blob/master/LICENSE
assertions go-render	https://github.com/smartystreets/assertions/internal/go-render/render	https://github.com/smartystreets/assertions/blob/master/internal/go-render/LICENSE
assertions oglmatchers	https://github.com/smartystreets/assertions/internal/oglematchers	https://github.com/smartystreets/assertions/blob/master/internal/oglematchers/LICENSE
assertions	https://github.com/smartystreets/assertions	https://github.com/smartystreets/assertions/blob/master/LICENSE.md
asttokens	https://github.com/gristlabs/asttokens	https://github.com/gristlabs/asttokens/blob/master/LICENSE
astunparse	https://github.com/simonpercivall/astunparse	https://github.com/simonpercivall/astunparse/blob/master/LICENSE
async-lru	https://github.com/aio-libs/async-lru	https://github.com/aio-libs/async-lru/blob/master/LICENSE
async-timeout	https://github.com/aio-libs/async-timeout	https://github.com/aio-libs/async-timeout/blob/master/LICENSE
atk-1.0	https://docs.gtk.org/atk/	https://gitlab.gnome.org/GNOME/atk/-/blob/master/COPYING
atomic	https://github.com/uber-go/atomic	https://github.com/uber-go/atomic/blob/master/LICENSE.txt

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
atomicwrites	https://github.com/untitaker/python-atomicwrites	https://github.com/untitaker/python-atomicwrites/blob/master/LICENSE
attrs	https://www.attrs.org/	https://github.com/python-attrs/attrs/blob/main/LICENSE
aws-sdk-go	https://github.com/aws/aws-sdk-go	https://github.com/aws/aws-sdk-go/blob/master/LICENSE.txt
babel	https://babel.pocoo.org/ OR https://github.com/python-babel/babel	https://github.com/python-babel/babel/blob/master/LICENSE OR https://babel.pocoo.org/en/latest/license.html#babel-license
Babel	https://github.com/python-babel/babel	https://github.com/python-babel/babel/blob/master/LICENSE
backcall	https://github.com/takluyver/backcall	https://github.com/takluyver/backcall/blob/master/LICENSE
backports	https://github.com/brandon-rhodes/backports	https://github.com/brandon-rhodes/backports/blob/master/LICENSE.TXT
backports.functools-lru-cache	https://github.com/jaraco/backports.functools_lru_cache	https://github.com/jaraco/backports.functools_lru_cache/blob/main/LICENSE
backports.zstd	https://github.com/rogdham/backports.zstd	https://github.com/rogdham/backports.zstd?tab=License-1-ov-file#readme
base62	https://github.com/kare/base62	https://github.com/kare/base62/blob/main/LICENSE
beautifulsoup4	https://www.crummy.com/software/BeautifulSoup/	https://www.crummy.com/software/BeautifulSoup/

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
billiard	https://github.com/celery/billiard	https://github.com/celery/billiard/blob/master/LICENSE.txt
biopython	https://biopython.org	https://github.com/biopython/biopython/blob/master/LICENSE.rst
biotite	https://github.com/biotite-dev/biotite/blob/master/README.rst	https://github.com/biotite-dev/biotite/blob/master/LICENSE.rst
bitarray	https://github.com/ilanschnell/bitarray	https://github.com/ilanschnell/bitarray/blob/master/LICENSE
bitset	https://github.com/willf/bitset	https://github.com/willf/bitset/blob/master/LICENSE
blas	https://www.netlib.org/blas/	https://www.netlib.org/blas/#_licensing
bleach	https://github.com/mozilla/bleach	https://github.com/mozilla/bleach/blob/main/LICENSE
blessings	https://github.com/erikrose/blessings	https://github.com/erikrose/blessings/blob/master/LICENSE
blinker	https://pythonhosted.org/blinker/	https://github.com/pallets-eco/blinker/blob/master/LICENSE
bokeh	https://github.com/bokeh/bokeh	https://github.com/bokeh/bokeh/blob/branch-4.0/LICENSE.txt
blosc	https://github.com/Blosc/python-blosc	https://github.com/Blosc/python-blosc/blob/master/LICENSE.txt
blosc2	https://github.com/Blosc/python-blosc2	https://github.com/Blosc/c-blosc2/blob/main/LICENSE.txt

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
boltons	https://github.com/mahmoud/boltons	https://github.com/mahmoud/boltons/blob/master/LICENSE
Boltz	https://boltz.bio	https://github.com/jwohlwend/boltz/blob/main/LICENSE
boost	https://www.boost.org/doc/libs/1_71_0/libs/python/doc/html/index.html	https://www.boost.org/LICENSE_1_0.txt
boost-cpp	https://www.boost.org/doc/libs/1_71_0/libs/python/doc/html/index.html	https://www.boost.org/LICENSE_1_0.txt
bootstrap-vue	https://github.com/bootstrap-vue/bootstrap-vue	https://github.com/bootstrap-vue/bootstrap-vue/raw/master/LICENSE
boto3	https://github.com/boto/boto3	https://github.com/boto/boto3/blob/develop/LICENSE
botocore	https://github.com/boto/botocore	https://github.com/boto/botocore/blob/develop/LICENSE.txt
Bottleneck	https://bottleneck.readthedocs.io/en/latest/index.html	https://bottleneck.readthedocs.io/en/latest/license.html
Brotli	https://github.com/google/brotli	https://github.com/google/brotli/blob/master/LICENSE
brotli-bin	https://github.com/google/brotli	https://github.com/google/brotli/blob/master/LICENSE
brotli-python	http://python-hyper.org/projects/brotlipy/en/latest/	https://github.com/python-hyper/brotlicffi/blob/main/LICENSE
brotlicffi	https://github.com/python-hyper/brotlicffi	https://github.com/python-hyper/brotlicffi?tab=MIT-1-ov-file#readme
brotlipy	https://github.com/python-hyper/brotlicffi	https://github.com/python-hyper/brotlicffi/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
bson	http://github.com/py-bson/bson	https://github.com/py-bson/bson/blob/master/LICENSE_APACHE
bytefmt	https://code.cloudfoundry.org/bytefmt	https://github.com/cloudfoundry/bytefmt/blob/master/LICENSE
bzip2	https://www.bzip.org	https://spdx.org/licenses/bzip2-1.0.6.html
c-ares	https://github.com/c-ares/c-ares	https://github.com/c-ares/c-ares/blob/main/LICENSE.md
ca-certificates	https://github.com/conda-forge/ca-certificates-feedsto	https://wiki.mozilla.org/CA/Included_Certificates
cached-property	https://github.com/pydanny/cached-property	https://github.com/pydanny/cached-property/blob/master/LICENSE
cachetools	https://github.com/tkem/cachetools/	https://github.com/tkem/cachetools/blob/master/LICENSE
cairo	https://pycairo.readthedocs.io/en/latest/	https://github.com/pygobject/pycairo/blob/master/COPYING
canvas	https://github.com/Automattic/node-canvas	https://github.com/Automattic/node-canvas
cctbx	https://github.com/cctbx/cctbx_project	https://github.com/cctbx/cctbx_project/blob/master/cctbx/LICENSE_2_0.txt
celery	https://github.com/celery/celery	https://github.com/celery/celery/blob/master/LICENSE
Cerberus	https://docs.python-cerberus.org/en/stable/	https://docs.python-cerberus.org/en/stable/license.html
certifi	https://certifiio.readthedocs.io/en/latest/	https://wiki.mozilla.org/index.php?title=CA/IncludedCertificates&redirect=no

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
cffi	https://github.com/python-cffi	https://github.com/cffi/cffi/blob/master/COPYRIGHT
cftime	https://pypi.org/project/cftime/	https://github.com/Unidata/cftime/blob/master/LICENSE
chardet	https://github.com/chardet/chardet	https://github.com/chardet/chardet/blob/master/LICENSE
charset-normalizer	https://github.com/ousret/charset_normalizer	https://github.com/Ousret/charset_normalizer/blob/master/LICENSE
ChEMBL Structure Pipeline	https://github.com/chembl/ChEMBL_Structure_Pipeline/tree/master	https://github.com/chembl/ChEMBL_Structure_Pipeline/blob/master/LICENSE
choreographer	https://github.com/mazpie/choreographer	https://github.com/mazpie/choreographer/blob/main/LICENSE
chunkreader	https://github.com/jackc/chunkreader/v2	https://github.com/jackc/chunkreader/blob/master/LICENSE
click	https://palletsprojects.com/p/click/	https://github.com/pallets/click/blob/main/LICENSE.rst
click-completion	https://github.com/click-contrib/click-completion	https://github.com/click-contrib/click-completion/blob/master/LICENSE
click-didyoumean	https://github.com/click-contrib/click-didyoumean	https://github.com/click-contrib/click-didyoumean/blob/master/LICENSE
click-plugins	https://github.com/click-contrib/click-plugins	https://github.com/click-contrib/click-plugins/blob/master/LICENSE.txt

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
click-repl	https://github.com/untitaker/click-repl	https://github.com/click-contrib/click-repl/blob/master/LICENSE
cloudpickle	https://github.com/cloudpipe/cloudpickle	https://github.com/cloudpipe/cloudpickle/blob/master/LICENSE
cmake	https://cmake.org/	https://cmake.org/licensing/
colorama	https://github.com/tartley/colorama	https://github.com/tartley/colorama/blob/master/LICENSE.txt
comm	https://github.com/ipython/comm	https://github.com/ipython/comm/blob/master/LICENSE
compose	https://github.com/docker/compose	https://github.com/docker/compose/blob/v2/LICENSE
conda-content-trust	https://github.com/conda/conda-content-trust	https://github.com/conda/conda-content-trust/blob/main/LICENSE
conda-libmamba-solver	https://github.com/conda/conda-libmamba-solv	https://github.com/conda/conda-libmamba-solv/blob/main/LICENSE
conda-pack	https://conda.github.io/conda-pack/	https://github.com/conda/conda-pack/blob/main/LICENSE.txt
conda-package-handling	https://github.com/conda/conda-package-handli	https://github.com/conda/conda-package-handli/blob/main/LICENSE
conda-package-streaming	https://anaconda.org/conda-forge/conda-package-stream	https://github.com/conda/conda-package-stream/blob/main/LICENSE
conda-token	https://anaconda.org/anaconda/conda-token	https://anaconda.org/anaconda/conda-token

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
conda_package_strear	https://github.com/conda/conda-package-streamtree/main/conda_package_streaming	https://github.com/conda/conda-package-streamtab=License-1-ov-file
confuse	https://github.com/beetbox/confuse	https://github.com/beetbox/confuse/blob/main/LICENSE
contourpy	https://github.com/contourpy/contourpy	https://github.com/contourpy/contourpy/blob/main/LICENSE
cpp-expected	https://github.com/TartanLlama/expected	https://github.com/TartanLlama/expected?tab=CC0-1.0-1-ov-file#readme
cpp-peglib	https://github.com/yhirose/cpp-peglib	https://github.com/yhirose/cpp-peglib/blob/master/LICENSE
cryptography	https://github.com/pyca/cryptography	https://github.com/pyca/cryptography/blob/main/LICENSE
cssselect2	https://github.com/Kozea/cssselect2/	https://github.com/Kozea/cssselect2/blob/master/LICENSE
cuda-bindings	https://nvidia.github.io/cuda-python/cuda-bindings/latest/install.html	https://github.com/NVIDIA/cuda-python/blob/main/cuda_bindings/LICENSE
cuda-pathfinder	https://nvidia.github.io/cuda-python/cuda-pathfinder/latest/	https://nvidia.github.io/cuda-python/cuda-pathfinder/latest/license.html
cudatoolkit	https://developer.nvidia.com/cuda-toolkit	https://docs.nvidia.com/cuda/eula/index.html
cuequivariance-ops-cu12	https://developer.nvidia.com/cuequivariance	https://github.com/NVIDIA/cuEquivariance/blob/main/LICENSES/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
cuequivariance-ops-torch-cu12	https://developer.nvidia.com/cuequivariance	https://github.com/NVIDIA/cuEquivariance/blob/main/LICENSES/LICENSE
cuequivariance-torch	https://docs.nvidia.com/cuda/cuequivariance/	https://github.com/NVIDIA/cuEquivariance/blob/main/LICENSES/LICENSE
cupy-cuda113	https://cupy.dev/	https://github.com/cupy/cupy/blob/main/LICENSE
cupy-cuda11x	https://cupy.dev/	https://github.com/cupy/cupy/?tab=MIT-1-ov-file#readme
curl	https://curl.se	https://curl.se/docs/copyright.html
cycler	https://github.com/matplotlib/cycler	https://github.com/matplotlib/cycler/blob/main/LICENSE
cyrus-sasl	https://github.com/cyrusimap/cyrus-sasl	https://github.com/cyrusimap/cyrus-sasl/blob/master/COPYING
Cython	https://cython.org	https://www.apache.org/licenses/LICENSE-2.0.html
d3	https://github.com/mbostock/d3	https://github.com/mbostock/d3/raw/master/LICENSE
dataclasses	https://github.com/ericvsmith/dataclasses	https://github.com/ericvsmith/dataclasses/blob/master/LICENSE.txt
ddsketch	http://github.com/datadog/sketches-py	https://github.com/DataDog/sketches-py/blob/master/LICENSE
debugpy	https://aka.ms/debugpy	https://github.com/microsoft/debugpy/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
decimal	https://github.com/shopspring/decimal	https://github.com/shopspring/decimal/blob/master/LICENSE
decorator	https://github.com/micheles/decorator	https://github.com/micheles/decorator/blob/master/LICENSE.txt
deepdiff	https://github.com/seperman/deepdiff	https://github.com/seperman/deepdiff/blob/master/LICENSE
deeptime	https://github.com/deeptime-ml/deeptime	https://github.com/deeptime-ml/deeptime/blob/main/LICENSE.txt
defusedxml	https://github.com/tiran/defusedxml	https://github.com/tiran/defusedxml/blob/main/LICENSE
dftd3	https://github.com/dftd3/tad-dftd3	https://github.com/dftd3/tad-dftd3/blob/main/LICENSE
dill	https://github.com/uqfoundation/dill	https://github.com/uqfoundation/dill/blob/master/LICENSE
distlib	https://distlib.readthedocs.io/en/latest/	https://github.com/pypa/distlib/blob/master/LICENSE.txt
distro	https://github.com/python-distro/distro	https://github.com/python-distro/distro/blob/master/LICENSE
Django	https://www.djangoproject.com/	https://github.com/django/django/blob/main/LICENSE
django-classy-tags	https://github.com/django-cms/django-classy-tags	https://github.com/django-cms/django-classy-tags/blob/master/LICENSE
django-cors-headers	https://github.com/adamchainz/django-cors-headers	https://github.com/adamchainz/django-cors-headers/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
django-csp	https://github.com/mozilla/django-csp	https://github.com/mozilla/django-csp/blob/main/LICENSE
django-extensions	https://github.com/django-extensions/django-extensions	https://github.com/django-extensions/django-extensions/blob/main/LICENSE
django-filter	https://github.com/carltongibson/django-filter/tree/master	https://github.com/carltongibson/django-filter/blob/main/LICENSE
django-redis	https://github.com/jazzband/django-redis	https://github.com/jazzband/django-redis/blob/master/LICENSE
django-taggit	https://github.com/jazzband/django-taggit	https://github.com/jazzband/django-taggit/blob/master/LICENSE
django-taggit-serializer	https://github.com/glemmaPaul/django-taggit-serializ	https://github.com/glemmaPaul/django-taggit-serializ blob/master/LICENSE
django-taggit-templatetags2	https://github.com/fizista/django-taggit-templat	https://github.com/fizista/django-taggit-templat blob/master/LICENSE.txt
djangorestframework	https://www.django-rest-framewor org/	https://github.com/encode/django-rest-framewor blob/master/LICENSE.md
dkh	https://psicode.org/psi4manual/master/dkh.html	https://github.com/psi4/dkh/blob/master/COPYING.LESSER
dm-tree	https://github.com/deepmind/tree	https://github.com/deepmind/tree/blob/master/LICENSE
docker-py	https://github.com/docker/docker-py/	https://github.com/docker/docker-py/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
docker-pycreds	https://github.com/shin-/dockerpy-creds	https://github.com/shin-/dockerpy-creds/blob/master/LICENSE
docopt	https://docopt.org	https://github.com/docopt/docopt/blob/master/LICENSE-MIT
docutils	https://docutils.sourceforge.io	https://docutils.sourceforge.io/COPYING.html
drf-dynamic-fields	https://github.com/dbrgn/drf-dynamic-fields	https://github.com/dbrgn/drf-dynamic-fields/blob/master/LICENSE
DuckDB	https://duckdb.org/	https://github.com/duckdb/duckdb?tab=MIT-1-ov-file#readme
edgembar	https://rutgerslbsr.gitlab.io/fe-toolkit/edgembar/index.html	https://gitlab.com/RutgersLBSR/fe-toolkit/-/blob/master/LICENSE
editdistance	https://github.com/roy-ht/editdistance	https://github.com/roy-ht/editdistance/blob/master/LICENSE
eigen	https://eigen.tuxfamily.org/index.php?title=Main_Page	https://eigen.tuxfamily.org/index.php?title=Main_Page#License
einops	https://github.com/arogozhnikov/einops	https://github.com/arogozhnikov/einops/blob/master/LICENSE
einx	https://github.com/fferflo/einx	https://github.com/fferflo/einx/blob/master/LICENSE
entrypoints	https://github.com/takluyver/entrypoints	https://github.com/takluyver/entrypoints/blob/master/LICENSE
errors	https://github.com/pkg/errors	https://github.com/pkg/errors/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
eslint-plugin	https://github.com/typescript-eslint/typescript-eslint	https://github.com/typescript-eslint/typescript-eslint/raw/master/LICENSE
et_xmlfile	https://foss.heptapod.net/openpyxl/et_xmlfile	https://foss.heptapod.net/openpyxl/et_xmlfile/-/blob/branch/default/LICENCE.rst
exceptiongroup	https://github.com/agronholm/exceptiongroup	https://github.com/agronholm/exceptiongroup/blob/main/LICENSE
execnet	https://execnet.readthedocs.io/en/latest/	https://github.com/pytest-dev/execnet/blob/master/LICENSE
executing	https://github.com/alexmojaki/executing	https://github.com/alexmojaki/executing/blob/master/LICENSE.txt
expat	https://libexpat.github.io	https://github.com/libexpat/libexpat/blob/master/expat/COPYING
fairscale	https://github.com/facebookresearch/fairscale	https://github.com/facebookresearch/fairscale/blob/main/LICENSE
fairseq	https://github.com/pytorch/fairseq	https://github.com/facebookresearch/fairseq/blob/main/LICENSE
fastjsonschema	https://github.com/horejsek/python-fastjsonschem	https://github.com/horejsek/python-fastjsonschem/blob/master/LICENSE
fastrlock	https://github.com/scoder/fastrlock	https://github.com/scoder/fastrlock/blob/master/LICENSE
fftw	https://www.fftw.org	commercial license

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
filebuffer	https://github.com/mattetti/filebuffer	https://github.com/mattetti/filebuffer/blob/master/LICENSE
filelock	https://py-filelock.readthedocs.io/en/latest/index.html	https://github.com/tox-dev/py-filelock/blob/main/LICENSE
finufft	https://github.com/flatironinstitute/finufft	https://github.com/flatironinstitute/finufft/blob/master/LICENSE
Flask	https://flask.palletsprojects.com/en/2.1.x/	https://flask.palletsprojects.com/en/2.1.x/license/
flatbuffers	https://google.github.io/flatbuffers/	https://github.com/google/flatbuffers/blob/master/LICENSE.txt
flexcache	https://github.com/hgrecco/flexcache	https://github.com/hgrecco/flexcache/blob/main/LICENSE
flexparser	https://github.com/hgrecco/flexparser	https://github.com/hgrecco/flexparser/blob/main/LICENSE
flit-core	https://github.com/pypa/flit	https://github.com/scientific-python/lazy_loader/blob/main/LICENSE.md
FLTK	https://www.fltk.org/index.php	https://www.fltk.org/COPYING.php
fmt	https://fmt.dev/latest/index.html	https://github.com/fmtlib/fmt/blob/master/LICENSE.rst
font-awesome	https://github.com/FortAwesome/Font-Awesome	https://github.com/FortAwesome/Font-Awesome
font-ttf-dejavu-sans-mono	https://dejavu-fonts.github.io	https://dejavu-fonts.github.io/License.html
font-ttf-inconsolata	https://fonts.google.com/specimen/Inconsolata	https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
font-ttf-source-code-pro	https://fonts.google.com/specimen/Source+Code+Pro	https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL
font-ttf-ubuntu	https://fonts.google.com/specimen/Ubuntu	https://ubuntu.com/legal/font-licence
fontconfig	https://www.freedesktop.org/wiki/Software/fontconfig/	https://gitlab.freedesktop.org/fontconfig/fontconfig/-/blob/main/COPYING
fonts-conda-ecosystem	https://anaconda.org/conda-forge/fonts-conda-ecosyster	https://docs.conda.io/en/latest/license.html
fonts-conda-forge	https://anaconda.org/conda-forge/fonts-conda-forge/	https://creativecommons.org/licenses/by/4.0/
fonttools	https://github.com/fonttools/fonttools	https://github.com/fonttools/fonttools/blob/main/LICENSE
forcebalance	https://simtk.org/home/forcebalance OR https://github.com/leeping/forcebalance	https://github.com/leeping/forcebalance/blob/master/LICENSE
fqdn	https://github.com/ypcrts/fqdn	https://github.com/ypcrts/fqdn?tab=MPL-2.0-1-ov-file#readme
freetype	https://freetype.org	https://freetype.org/license.html
freetype-py	https://github.com/rougier/freetype-py	https://github.com/rougier/freetype-py?tab=License-1-ov-file#readme
fribidi	https://github.com/fribidi/fribidi	https://github.com/fribidi/fribidi/blob/master/COPYING
frozendict	https://github.com/Marco-Sulla/python-frozendict	https://github.com/Marco-Sulla/python-frozendict/blob/master/LICENSE.txt
frozenlist	https://github.com/aio-libs/frozenlist	https://github.com/aio-libs/frozenlist/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
fsmlite	https://github.com/tkem/fsmlite	https://github.com/tkem/fsmlite/blob/master/LICENSE
fsspec	https://github.com/fsspec/filesystem_spec	https://github.com/fsspec/filesystem_spec/blob/master/LICENSE
funcy	https://github.com/Suor/funcy	https://github.com/Suor/funcy/blob/master/LICENSE
gast	https://github.com/serge-sans-paille/gast/	https://github.com/serge-sans-paille/gast/blob/master/LICENSE
gau2grid	https://github.com/dgasmith/gau2grid	https://github.com/dgasmith/gau2grid/blob/master/LICENSE
gax-go	https://github.com/googleapis/gax-go/v2	https://github.com/googleapis/gax-go/blob/main/LICENSE
gdk-pixbuf	https://gitlab.gnome.org/GNOME/gdk-pixbuf	https://gitlab.gnome.org/GNOME/gdk-pixbuf/-/blob/master/COPYING
gemmi	https://gemmi.readthedocs.io/en/latest/	https://github.com/project-gemmi/gemmi/blob/master/LICENSE.txt
genproto	https://google.golang.org/genproto/googleapis	https://github.com/googleapis/go-genproto/blob/main/LICENSE
geometric	https://openbase.com/python/geometric	https://openbase.com/python/geometric
giflib	https://giflib.sourceforge.net	https://github.com/mirrorer/giflib/blob/master/COPYING
gitdb	https://github.com/gitpython-developers/gitdb	https://github.com/gitpython-developers/gitdb/blob/master/LICENSE
GitPython	https://github.com/gitpython-developers/GitPython	https://github.com/gitpython-developers/GitPython/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
glib	https://docs.gtk.org/glib/	https://gitlab.gnome.org/GNOME/glib/-/blob/main/LICENSES/LGPL-2.1-or-later.txt
GLM Library	https://github.com/g-truc/glm	https://github.com/g-truc/glm/blob/master/manual.md#section0
gls	https://github.com/jtolds/gls	https://github.com/jtolds/gls/blob/master/LICENSE
gmpy2	https://github.com/aleaxit/gmpy	https://github.com/gmpy2/gmpy2/blob/master/COPYING.LESSER
go-cleanhttp	https://github.com/hashicorp/go-cleanhttp	https://github.com/hashicorp/go-cleanhttp/blob/master/LICENSE
go-connections	https://github.com/docker/go-connections	https://github.com/docker/go-connections/blob/master/LICENSE
go-cpio	https://github.com/cavaliercoder/go-cpio	https://github.com/cavaliercoder/go-cpio/blob/master/LICENSE
go-getter	https://github.com/hashicorp/go-getter	https://github.com/hashicorp/go-getter/blob/master/LICENSE
go-homedir	https://github.com/mitchellh/go-homedir	https://github.com/mitchellh/go-homedir/blob/master/LICENSE
go-ini	https://github.com/go-ini/ini	https://github.com/go-ini/ini/blob/master/LICENSE
go-jmespath	https://github.com/jmespath/go-jmespath	https://github.com/jmespath/go-jmespath/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
go-junit-report	https://github.com/jstemmer/go-junit-report	https://github.com/jstemmer/go-junit-report/blob/master/LICENSE
go-netrc	https://github.com/bgentry/go-netrc/netrc	https://github.com/bgentry/go-netrc/blob/master/LICENSE
go-ole	https://github.com/go-ole/go-ole	https://github.com/go-ole/go-ole/blob/master/LICENSE
go-pg	https://github.com/go-pg/pg	https://github.com/go-pg/pg/blob/master/LICENSE
go-redis	https://github.com/go-redis/redis/v8	https://github.com/go-redis/redis/blob/master/LICENSE
go-rendezvous	https://github.com/dgryski/go-rendezvous	https://github.com/dgryski/go-rendezvous/blob/master/LICENSE
go-safetemp	https://github.com/hashicorp/go-safetemp	https://github.com/hashicorp/go-safetemp/blob/master/LICENSE
go-sysconf	https://github.com/tklauser/go-sysconf	https://github.com/tklauser/go-sysconf/blob/master/LICENSE
go-testing-interface	https://github.com/mitchellh/go-testing-interface	https://github.com/mitchellh/go-testing-interface/blob/master/LICENSE
go-units	https://github.com/docker/go-units	https://github.com/docker/go-units/blob/master/LICENSE
go-version	https://github.com/hashicorp/go-version	https://github.com/hashicorp/go-version/blob/master/LICENSE
go-zglob	https://github.com/mattn/go-zglob	https://github.com/mattn/go-zglob/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
go.opencensus	https://go.opencensus.io	https://github.com/census-instrumentatio opencensus-go/blob/master/LICENSE
gobrake.v2	https://gopkg.in/airbrake/gobrake.v2	https://github.com/airbrake/gobrake/blob/v2.0.9/LICENSE
goconvey	https://github.com/smartystreets/goconvey	https://github.com/smartystreets/goconvey/blob/master/LICENSE.md
golden-layout	https://github.com/deepstreamIO/golden-layout	https://github.com/deepstreamIO/golden-layout/raw/master/LICENSE
google-auth	https://google-auth.readthedocs.io/en/master/	https://github.com/GoogleCloudPlatform google-auth-library-p blob/master/LICENSE
google-auth-oauthlib	https://github.com/googleapis/google-auth-library-p	https://github.com/googleapis/google-auth-library-p blob/main/LICENSE
google-cloud-go	https://cloud.google.com/go	https://github.com/googleapis/google-cloud-go/blob/main/LICENSE
google-cloud-go/storage	https://cloud.google.com/go/storage	https://github.com/googleapis/google-cloud-go/blob/main/LICENSE
google-pasta	https://github.com/google/pasta	https://github.com/google/pasta/blob/master/LICENSE
google.golang.org/api	https://google.golang.org/api	https://github.com/googleapis/google-api-go-client/blob/main/LICENSE
gopsutil	https://github.com/shirou/gopsutil	https://github.com/shirou/gopsutil/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
gorilla websocket	https://github.com/gorilla/websocket	https://github.com/gorilla/websocket/blob/master/LICENSE
gpytorch	https://gpytorch.ai	https://github.com/cornellius-gp/gpytorch?tab=MIT-1-ov-file#readme
graphite2	https://github.com/silnrsi/graphite	https://github.com/silnrsi/graphite/blob/master/LICENSE
graphviz	https://graphviz.org/	https://graphviz.org/license/
greenlet	https://greenlet.readthedocs.io/en/latest/	https://github.com/python-greenlet/greenlet/blob/master/LICENSE
groupcache	https://github.com/golang/groupcache	https://github.com/golang/groupcache/blob/master/LICENSE
grpc	https://google.golang.org/grpc	https://github.com/grpc/grpc-go/blob/master/LICENSE
grpc-cpp	https://github.com/grpc/grpc	https://github.com/grpc/grpc/blob/master/LICENSE
grpcio	https://github.com/grpc/grpc.io/blob/main/LICENSE	https://github.com/grpc/grpc.io/blob/main/LICENSE
gtk2	https://gitlab.gnome.org/GNOME/gtk	https://gitlab.gnome.org/GNOME/gtk/-/blob/main/COPYING
gts	https://sourceforge.net/projects/gts/	https://www.gnu.org/licenses/lgpl-3.0.html
h11	https://github.com/python-hyper/h11	https://github.com/python-hyper/h11/blob/master/LICENSE.txt
h2	https://github.com/python-hyper/h2	https://github.com/python-hyper/h2?tab=MIT-1-ov-file#readme

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
h5py	https://www.h5py.org	https://docs.h5py.org/en/stable/licenses.html
harfbuzz	https://github.com/harfbuzz/harfbuzz	https://github.com/harfbuzz/harfbuzz/blob/main/COPYING
hdbscan	https://hdbscan.readthedocs.io/en/latest/	https://github.com/scikit-learn-contrib/hdbscan/blob/master/LICENSE
hdf4	https://www.hdfgroup.org/solutions/hdf4/	https://github.com/HDFGroup/hdf4/blob/master/COPYING
hdf5	https://www.hdfgroup.org/solutions/hdf5/	https://github.com/HDFGroup/hdf5/blob/master/COPYING
he	https://github.com/mathiasbynens/he	https://github.com/mathiasbynens/he/raw/master/LICENSE-MIT.txt
hpack	https://github.com/python-hyper/hpack	https://github.com/python-hyper/hpack?tab=MIT-1-ov-file#readme
html-loader	https://github.com/webpack-contrib/html-loader	https://github.com/webpack-contrib/html-loader/raw/master/LICENSE
html5lib	https://github.com/html5lib/html5lib-python	https://github.com/html5lib/html5lib-python/blob/master/LICENSE
htslib	https://www.htslib.org	https://github.com/samtools/htslib/blob/develop/LICENSE
httpcore	https://www.encode.io/httpcore/	https://github.com/encode/httpcore/blob/master/LICENSE.md
httpx	https://github.com/encode/httpx	https://github.com/encode/httpx/blob/master/LICENSE.md

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
humanize	https://github.com/jmoiron/humanize	https://github.com/jmoiron/humanize/blob/master/LICENCE
hydra-core	https://github.com/facebookresearch/hydra	https://github.com/facebookresearch/hydra/blob/main/LICENSE
hyperframe	https://github.com/python-hyper/hyperframe	https://github.com/python-hyper/hyperframe?tab=MIT-1-ov-file#readme
icu	https://github.com/unicode-org/icu	https://github.com/unicode-org/icu/blob/main/icu4c/LICENSE
idna	https://github.com/kjd/idna	https://github.com/kjd/idna/blob/master/LICENSE.md
ihm	https://github.com/ihmwg/python-ihm	https://github.com/ihmwg/python-ihm/blob/main/LICENSE
imageio	https://github.com/imageio/imageio	https://github.com/imageio/imageio/blob/master/LICENSE
imagesize	https://github.com/shibukawa/imagesize_py	https://github.com/shibukawa/imagesize_py/blob/master/LICENSE.rst
ImmuneBuilder	https://github.com/oxpig/ImmuneBuilder/tree/main	https://github.com/oxpig/ImmuneBuilder/blob/main/LICENSE
importlib-metadata	https://github.com/python/importlib_metadata	https://github.com/python/importlib_metadata/blob/main/LICENSE
importlib_metadata	https://github.com/python/importlib_metadata	https://github.com/python/importlib_metadata/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
importlib_resources	https://github.com/python/importlib_resources	https://github.com/python/importlib_resources/blob/main/LICENSE
InChI	https://iupac.org/who-we-are/divisions/division-details/inchi/	https://iupac.org/wp-content/uploads/2021/06/iupac-inchi-license_2020.pdf
inflection	https://github.com/jinzhu/inflection	https://github.com/jinzhu/inflection/blob/master/LICENSE
ini.v1	https://gopkg.in/ini.v1	https://github.com/go-ini/ini/blob/v1.66.2/LICENSE
iniconfig	https://github.com/pytest-dev/iniconfig	https://github.com/pytest-dev/iniconfig/blob/main/LICENSE
innersvg-polyfill	https://github.com/dnozay/innersvg-polyfill	https://github.com/dnozay/innersvg-polyfill/raw/master/LICENSE
intel-openmp	https://github.com/hermitcore/libomp_oss	https://github.com/hermitcore/libomp_oss/blob/hermit/LICENSE.txt
ipykernel	https://ipython.org	https://ipython.org
ipython	https://ipython.org	https://github.com/ipython/ipython/blob/master/LICENSE
ipython-genutils	http://ipython.org	https://github.com/ipython/ipython_genutils/blob/master/COPYING.md
ipython_pygments_le	https://github.com/ipython/ipython-pygments-lex	https://github.com/ipython/ipython-pygments-lex blob/main/LICENSE
ipywidgets	http://jupyter.org	https://github.com/jupyter-widgets/ipywidgets/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
isodate	https://github.com/gweis/isodate/	https://github.com/gweis/isodate/blob/master/LICENSE
isoduration	https://github.com/bolsote/isoduration	https://github.com/bolsote/isoduration?tab=ISC-1-ov-file#readme
itsdangerous	https://palletsprojects.com/p/itsdangerous/	https://itsdangerous.palletsprojects.com/en/2.1.x/license/
jax	https://github.com/google/jax	https://github.com/google/jax/blob/main/LICENSE
jaxlib	https://github.com/google/jax	https://github.com/google/jax/blob/main/LICENSE
jaxtyping	https://github.com/patrick-kidger/jaxtyping	https://github.com/patrick-kidger/jaxtyping?tab=License-1-ov-file#readme
jedi	https://jedi.readthedocs.io/en/latest/	https://github.com/davidhalter/jedi/blob/master/LICENSE.txt
Jinja2	https://palletsprojects.com/p/jinja/	https://github.com/pallets/jinja/blob/main/LICENSE.rst
jmespath	https://github.com/jmespath/jmespath.py	https://github.com/jmespath/jmespath.py/blob/develop/LICENSE.txt
joblib	https://joblib.readthedocs.io/en/latest/	https://github.com/joblib/joblib/blob/master/LICENSE.txt
jpeg	https://www.ijg.org	https://www.ijg.org/files/README
json5	https://github.com/dpranke/pyjson5	https://github.com/dpranke/pyjson5/blob/main/LICENSE
jsonfield	https://github.com/dmkoch/django-jsonfield/	https://github.com/rpkilby/jsonfield/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
jsonpatch	https://github.com/stefankoegl/python-json-patch	https://github.com/stefankoegl/python-json-patch/blob/master/LICENSE
jsonpickle	https://github.com/jsonpickle/jsonpickle	https://github.com/jsonpickle/jsonpickle/blob/main/LICENSE
jsonpointer	https://github.com/stefankoegl/python-json-pointer	https://github.com/stefankoegl/python-json-pointer/blob/master/LICENSE.txt
jsonschema	https://github.com/python-jsonschema/jsonschema	https://github.com/python-jsonschema/jsonschema/blob/master/COPYING
jsonschema-specifications	https://github.com/python-jsonschema/jsonschema-specificat	https://github.com/python-jsonschema/jsonschema-specificat blob/main/COPYING
jstat	https://github.com/jstat/jstat	https://www.opensource.org/licenses/mit-license.php
jupyter-client	https://jupyter.org	https://github.com/jupyter/jupyter_client/blob/main/COPYING.md
jupyter-core	https://jupyter.org	https://github.com/jupyter/jupyter_core/blob/master/COPYING.md
jupyter-events	https://github.com/jupyter/jupyter_events	https://github.com/jupyter/jupyter_events/blob/main/LICENSE
jupyter-lsp	https://github.com/jupyter-lsp/jupyterlab-lsp	https://github.com/jupyter-lsp/jupyterlab-lsp
jupyter-serverhttp://jupyter.or	https://github.com/jupyter-server/jupyter_server/blob/main/LICENSE	
jupyter_client	http://jupyter.org	https://github.com/jupyter/jupyter_client/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
jupyter_core	http://jupyter.org	https://github.com/jupyter/jupyter_core/blob/master/LICENSE
jupyter_server	https://github.com/jupyter-server/jupyter_server	https://github.com/jupyter-server/jupyter_server/blob/main/LICENSE
jupyter_server_termin	https://github.com/jupyter-server/jupyter_server_terminals	https://github.com/jupyter-server/jupyter_server_terminals/blob/main/LICENSE
jupyterlab	https://github.com/jupyterlab/jupyterlab	https://github.com/jupyterlab/jupyterlab/blob/main/LICENSE
jupyterlab-pygments	http://jupyter.org	https://github.com/jupyterlab/jupyterlab_pygments/blob/main/LICENSE
jupyterlab-widgets	http://jupyter.org	https://github.com/jupyter-widgets/ipywidgets/blob/master/LICENSE
jupyter-lab_pygments	https://github.com/jupyterlab/jupyterlab_pygments	https://github.com/jupyterlab/jupyterlab_pygments?tab=BSD-3-Clause-1-ov-file#readme
jupyterlab_server	https://github.com/jupyterlab/jupyterlab_server	https://github.com/jupyterlab/jupyterlab_server/blob/main/LICENSE
jupyterlab_widgets	https://github.com/jupyter-widgets/ipywidgets	https://github.com/jupyter-widgets/ipywidgets?tab=BSD-3-Clause-1-ov-file#readme
kaleido	https://github.com/plotly/Kaleido	https://github.com/plotly/Kaleido/blob/master/LICENSE.txt
keras	https://github.com/keras-team/keras	https://github.com/keras-team/keras/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
Keras-Preprocessing	https://github.com/keras-team/keras-preprocessing	https://github.com/keras-team/keras-preprocessing/blob/master/LICENSE
keras-tuner	https://github.com/keras-team/keras-tuner	https://github.com/keras-team/keras-tuner/blob/master/LICENSE
keyring	https://github.com/jaraco/keyring	https://github.com/jaraco/keyring/blob/main/LICENSE
keyutils	https://github.com/sassoftware/python-keyutils	https://github.com/sassoftware/python-keyutils/blob/master/LICENSE
kiwisolver	https://kiwisolver.readthedocs.io/en/latest/	https://github.com/nucleic/kiwi/blob/main/LICENSE
kombu	https://kombu.readthedocs.io	https://github.com/celery/kombu/blob/master/LICENSE
krb5	https://web.mit.edu/kerberos/	https://github.com/krb5/krb5/blob/master/src/lib/gssapi/LICENSE
kt-legacy	https://github.com/haifeng-jin/kt-legacy	https://github.com/haifeng-jin/kt-legacy/blob/main/LICENSE
lark	https://github.com/lark-parser/lark	https://github.com/lark-parser/lark?tab=MIT-1-ov-file#readme
lazy_loader	https://github.com/scientific-python/lazy_loader	https://github.com/scientific-python/lazy_loader/blob/main/LICENSE.md
lcms2	https://www.littlecms.com	https://github.com/sk1project/python-lcms2/blob/master/LICENSE
legacy-cgi	https://github.com/jackrosenthal/legacy-cgi	https://github.com/jackrosenthal/legacy-cgi?tab=License-1-ov-file#readme

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
lerc	https://github.com/Esri/lerc	https://github.com/Esri/lerc/blob/master/LICENSE
libarchive	http://www.libarchive.org	https://raw.githubusercontent.com/libarchive/libarchive/master/COPYING
libblas	https://www.netlib.org/blas/	https://www.netlib.org/blas/#_licensing
libbrotlicommon	https://github.com/google/brotli	https://github.com/google/brotli/blob/master/LICENSE
libbrotlidec	https://github.com/google/brotli	https://github.com/google/brotli/blob/master/LICENSE
libbrotlienc	https://github.com/google/brotli	https://github.com/google/brotli/blob/master/LICENSE
libcblas	https://anaconda.org/conda-forge/libcblas	N/A
libclang	https://github.com/sighingnow/libclang	https://github.com/sighingnow/libclang/blob/master/LICENSE.TXT
libcurl	https://curl.se/libcurl/	https://curl.se/docs/copyright.html
libcxx	https://github.com/llvm-mirror/libcxx	https://github.com/llvm-mirror/libcxx/blob/master/LICENSE.TXT
libdb	https://www.oracle.com/technology/software/products/berkeley-db/index.html	https://github.com/berkeleydb/libdb/blob/master/LEGAL
libdeflate	https://github.com/ebiggers/libdeflate	https://github.com/ebiggers/libdeflate/blob/master/COPYING
libedit	https://thrysoee.dk/editline/	http://www.netbsd.org/Goals/redistribution.html
libev	https://software.schmorp.de/pkg/libev.html	https://software.schmorp.de/pkg/libev.html

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
libexpat	https://libexpat.github.io/	https://github.com/libexpat/libexpat?tab=MIT-1-ov-file#readme
libffi	https://github.com/libffi/libffi	https://github.com/libffi/libffi/blob/master/LICENSE
libgcc	https://gcc.gnu.org/onlinedocs/gccint/Libgcc.html	https://gcc.gnu.org/onlinedocs/libstdc++/manual/license.html
libgcrypt	https://gnupg.org/software/index.html	https://git.gnupg.org/cgi-bin/gitweb.cgi?p=libgcrypt.git;a=blob;f=COPYING.LIB;hb=HEAD
libgd	https://libgd.github.io	https://libgd.github.io/manuals/2.2.1/files/license-txt.html
libglib	https://github.com/PolMine/libglib	https://github.com/PolMine/libglib/blob/master/include/glib-2.0/glib.h
libiconv	https://www.gnu.org/software/libiconv/	https://www.gnu.org/software/libiconv/#downloading
libint	https://tinyurl.com/yvw97wbw	https://tinyurl.com/yvw97wbw
liblapack	http://www.netlib.org/lapack/	https://www.netlib.org/lapack/LICENSE.txt
liblapacke	https://anaconda.org/conda-forge/liblapacke	https://www.netlib.org/lapack/LICENSE.txt
liblzma	https://tukaani.org/xz/	https://git.tukaani.org/?p=xz.git;a=blob;f=COPYING.0BSD
libmamba	https://medium.com/@QuantStack/open-software-packag	https://github.com/mamba-org/mamba/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
libmambapy	https://www.anaconda.com/blog/a-faster-conda-for-a-g	https://github.com/conda/conda-libmamba-solv/blob/main/LICENSE
libnetcdf	https://www.unidata.ucar.edu/software/netcdf/	https://www.unidata.ucar.edu/software/netcdf/copyright.html
libnghttp2	https://github.com/nghttp2/nghttp2	https://github.com/nghttp2/nghttp2/blob/master/COPYING
libnsl	https://github.com/thkukuk/libnsl	https://gcc.gnu.org/onlinedocs/libstdc++/manual/license.html
libopenblas	https://www.openblas.net/	https://github.com/xianyi/OpenBLAS/blob/develop/LICENSE
libpng	https://www.libpng.org/pub/png/libpng.html	https://www.libpng.org/pub/png/src/libpng-LICENSE.txt
libpq	https://www.postgresql.org/	https://www.postgresql.org/about/licence/
librsvg	https://gitlab.gnome.org/GNOME/librsvg	https://gitlab.gnome.org/GNOME/librsvg/-/blob/main/COPYING.LIB
libsolv	https://github.com/openSUSE/libsolv	https://github.com/openSUSE/libsolv/blob/master/LICENSE.BSD
libsqlite	https://github.com/mackyle/sqlite	https://github.com/mackyle/sqlite?tab=License-1-ov-file#readme
libssh2	https://github.com/libssh2/libssh2	https://github.com/libssh2/libssh2/blob/master/COPYING
libstdcxx	https://gcc.gnu.org/	https://gcc.gnu.org/onlinedocs/libstdc++/manual/license.html

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
libtiff	https://www.libtiff.org/	https://www.libtiff.org/misc.html
libtrust	https://github.com/docker/libtrust	https://github.com/docker/libtrust/blob/master/LICENSE
libuuid	https://sourceforge.net/projects/libuuid/	https://sourceforge.net/p/libuuid/code/ci/master/tree/COPYING
libuv	https://github.com/libuv/libuv	https://github.com/libuv/libuv/blob/v1.x/LICENSE
libwebp	https://chromium.googlesource.com/?format=HTML	https://chromium.googlesource.com/chromium/src/third_party/+/refs/heads/main/libwebp/LICENSE
libwebp-base	https://chromium.googlesource.com/?format=HTML	https://chromium.googlesource.com/chromium/src/third_party/+/refs/heads/main/libwebp/LICENSE
libxc	https://www.tddft.org/programs/libxc/	https://gitlab.com/libxc/libxc/-/blob/master/COPYING
libxcb	https://xcb.freedesktop.org	https://github.com/freedesktop/xcb-libxcb/blob/master/COPYING
libxcrypt	https://github.com/besser82/libxcrypt	https://github.com/besser82/libxcrypt?tab=LGPL-2.1-1-ov-file#readme
libxml2	https://git.gnome.org/browse/libxml2/	https://gitlab.gnome.org/GNOME/libxml2/-/blob/master/Copyright
libxmlsec1	https://github.com/lsh123/xmlsec	https://github.com/lsh123/xmlsec/blob/master/README
libxslt	https://gitlab.gnome.org/GNOME/libxslt/-/wikis/home	https://gitlab.gnome.org/GNOME/libxslt/-/blob/master/Copyright

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
libzlib	https://zlib.net	https://zlib.net/zlib_license.html
lightning-utilities	https://github.com/Lightning-AI/utilities	https://github.com/Lightning-AI/utilities?tab=Apache-2.0-1-ov-file#readme
lime	https://github.com/marcotcr/lime	https://github.com/marcotcr/lime/blob/master/LICENSE
linear-operator	https://github.com/cornellius-gp/linear_operator	https://github.com/cornellius-gp/linear_operator?tab=MIT-1-ov-file#readme
lit	http://llvm.org	https://releases.llvm.org/14.0.0/LICENSE.TXT
llvm-openmp	https://github.com/llvm-mirror/openmp	https://github.com/llvm-mirror/openmp/blob/master/LICENSE.txt
llvmlite	http://llvmlite.readthedocs.io	https://github.com/numba/llvmlite/blob/main/LICENSE
loader-utils	https://github.com/webpack/loader-utils	https://github.com/webpack/loader-utils/raw/master/LICENSE
logistro	https://github.com/geopozo/logistro	https://github.com/geopozo/logistro?tab=MIT-1-ov-file#readme
logomaker	https://logomaker.readthedocs.io/en/latest/	https://github.com/jbkinney/logomaker/blob/master/LICENSE
logrus	https://github.com/sirupsen/logrus	https://github.com/sirupsen/logrus/blob/master/LICENSE
logrus-airbrake-hook.v2	https://gopkg.in/gemnasium/logrus-airbrake-hook.v2	https://github.com/gemnasium/logrus-airbrake-hook/blob/v2.1.2/LICENSE
lxml	https://lxml.de	https://lxml.de/index.html#license

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
lz4-c	https://www.lz4.org/	https://github.com/lz4/lz4/blob/dev/LICENSE
lzo	https://www.oberhumer.com/opensource/lzo/	https://www.oberhumer.com/opensource/gpl.html
mamba	https://github.com/mamba-org/mamba	https://github.com/mamba-org/mamba?tab=BSD-3-Clause-1-ov-f readme
markdown	https://github.com/evilstreak/markdown-js	https://www.opensource.org/licenses/mit-license.php
markdown-it-py	https://github.com/executablebooks/markdown-it-py	https://github.com/executablebooks/markdown-it-py/blob/master/LICENSE
markupsafe	https://github.com/pallets/markupsafe/tree/main/src/markupsafe	https://github.com/pallets/markupsafe/tree/main?tab=BSD-3-Clause-1-ov-f
MarkupSafe	https://palletsprojects.com/p/markupsafe/	https://github.com/pallets/markupsafe/blob/main/LICENSE.rst
mashumaro	https://github.com/Fatal1ty/mashumaro	https://github.com/Fatal1ty/mashumaro
matplotlib	https://matplotlib.org	https://matplotlib.org/stable/users/project/license.html
matplotlib-base	https://matplotlib.org	https://matplotlib.org/stable/users/project/license.html
matplotlib-inline	https://github.com/ipython/matplotlib-inline	https://github.com/ipython/matplotlib-inline/blob/master/LICENSE
mda-xdrlib	https://github.com/MDAnalysis/mda-xdrlib	https://github.com/MDAnalysis/mda-xdrlib/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
mda_xdrlib	https://github.com/MDAnalysis/mda-xdrlib	https://github.com/MDAnalysis/mda-xdrlib?tab=License-1-ov-file#readme
mdtraj	https://www.mdtraj.org/	https://www.mdtraj.org/1.9.8.dev0/index.html#license
mdurl	https://github.com/markdown-it/mdurl	https://github.com/markdown-it/mdurl/blob/master/LICENSE
menuinst	https://github.com/conda/menuinst	https://github.com/conda/menuinst/blob/main/LICENSE.txt
mergo	https://github.com/imdario/mergo	https://github.com/imdario/mergo/blob/master/LICENSE
mistune	https://github.com/lepture/mistune	https://github.com/lepture/mistune/blob/master/LICENSE
mkl	https://github.com/rust-math/intel-mkl-src	https://github.com/rust-math/intel-mkl-src/blob/master/License.txt
mkl-fft	https://github.com/IntelPython/mkl_fft	https://github.com/IntelPython/mkl_fft/blob/master/LICENSE.txt
mkl-random	https://github.com/IntelPython/mkl_random	https://github.com/IntelPython/mkl_random/blob/master/LICENSE.txt
mkl-service	https://github.com/IntelPython/mkl-service	https://github.com/IntelPython/mkl-service/blob/master/LICENSE.txt
mkl_fft	http://github.com/IntelPython/mkl_fft	https://github.com/IntelPython/mkl_fft?tab=BSD-3-Clause-1-ov-file#readme

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
mkl_random	http://github.com/IntelPython/mkl_random	https://github.com/IntelPython/mkl_random?tab=BSD-3-Clause-1-ov-f readme
ml-dtypes	https://github.com/jax-ml/ml_dtypes	https://github.com/jax-ml/ml_dtypes/blob/main/LICENSE
ml_dtypes	https://github.com/jax-ml/ml_dtypes	https://github.com/jax-ml/ml_dtypes/blob/main/LICENSE
modelcif	https://github.com/ihmwg/python-modelcif	https://github.com/ihmwg/python-modelcif/blob/main/LICENSE
molecupy	https://molecupy.readthedocs.io/en/latest/	https://github.com/samirelanduk/atomium/blob/master/LICENSE
moment	https://github.com/moment/moment	https://github.com/moment/moment/raw/master/LICENSE
moment-precise-range-plugin	https://github.com/codebox/moment-precise-range	https://github.com/codebox/moment-precise-range raw/master/LICENSE.md
more-itertools	https://github.com/more-itertools/more-itertools	https://github.com/more-itertools/more-itertools/blob/master/LICENSE
mpiplus	https://github.com/choderalab/mpiplus	https://github.com/choderalab/mpiplus/blob/main/LICENSE
mpmath	http://mpmath.org/	https://github.com/mpmath/mpmath/blob/master/LICENSE
mrcfile	https://github.com/ccpem/mrcfile	https://github.com/ccpem/mrcfile/blob/master/LICENSE.txt

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
msgpack	https://msgpack.org/	https://github.com/msgpack/msgpack-python/blob/main/COPYING
multidict	https://github.com/aio-libs/multidict	https://github.com/aio-libs/multidict/blob/master/LICENSE
multierr	https://go.uber.org/multierr	https://github.com/uber-go/multierr/blob/master/LICENSE.txt
multiprocess	https://github.com/uqfoundation/multiprocess	https://github.com/uqfoundation/multiprocess/blob/master/LICENSE
munkres	https://software.clapper.org/munkres/	https://github.com/bmc/munkres/blob/master/LICENSE.md
myesui uuid	https://github.com/myesui/uuid	https://pkg.go.dev/github.com/myesui/uuid?tab=licenses
namex	https://github.com/bcgov/namex	https://github.com/bcgov/namex/blob/main/LICENSE
narwhals	https://github.com/narwhals-dev/narwhals	https://github.com/narwhals-dev/narwhals?tab=MIT-1-ov-file#readme
nbclassic	http://jupyter.org	https://github.com/jupyter/nbclassic/blob/main/LICENSE
nbclient	https://jupyter.org	https://github.com/jupyter/nbclient/blob/main/LICENSE
nbconvert	https://jupyter.org	https://github.com/jupyter/nbconvert/blob/main/LICENSE
nbformat	http://jupyter.org	https://github.com/jupyter/nbformat/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
ncurses	https://invisible-island.net/ncurses/	https://invisible-island.net/ncurses/ncurses-license.html
ndfes	https://rutgerslbsr.gitlab.io/fe-toolkit/	https://gitlab.com/RutgersLBSR/fe-toolkit/-/blob/master/LICENSE
ndindex	https://quansight-labs.github.io/ndindex/	https://github.com/Quansight-Labs/ndindex?tab=MIT-1-ov-file#readme
nest-asyncio	https://github.com/erdewit/nest_asyncio	https://github.com/erdewit/nest_asyncio/blob/master/LICENSE
nest_asyncio	https://github.com/erdewit/nest_asyncio	https://github.com/erdewit/nest_asyncio?tab=BSD-2-Clause-1-ov-f readme
netcdf-fortran	https://www.unidata.ucar.edu/software/netcdf/	https://www.unidata.ucar.edu/software/netcdf/copyright.html
netCDF4	http://github.com/Unidata/netcdf4-python	https://github.com/Unidata/netcdf4-python/blob/master/LICENSE
nettle	https://git.lysator.liu.se/nettle/nettle	https://git.lysator.liu.se/nettle/nettle/-/blob/master/AUTHORS?ref_type=heads
networkx	https://networkx.org	https://github.com/networkx/networkx/blob/main/LICENSE.txt
nfpm	https://github.com/goreleaser/nfpm	https://github.com/goreleaser/nfpm/blob/master/LICENSE.md
ng-tags-input	https://github.com/mbenford/ngTagsInput	https://github.com/mbenford/ngTagsInput
ng-toast	https://github.com/tameraydin/ngToast	https://github.com/tameraydin/ngToast

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
ngdraggable	https://github.com/fatlinesofcode/ngDraggable	https://github.com/fatlinesofcode/ngDraggable/raw/master/LICENSE.txt
ngVue	https://github.com/ngVue/ngVue	https://github.com/ngVue/ngVue/raw/master/LICENSE
nh3	https://nh3.readthedocs.io/en/latest/	https://github.com/messense/nh3/blob/main/LICENSE
nlohmann_json	https://github.com/nlohmann/json	https://github.com/nlohmann/json?tab=MIT-1-ov-file#readme
nlopt	https://nlopt.readthedocs.io/en/latest/NLopt_Python_Reference/	https://nlopt.readthedocs.io/en/latest/NLopt_License_and_Copyright/
nodejs	https://nodejs.org/en/	https://raw.githubusercontent.com/nodejs/node/master/LICENSE
nomkl	https://github.com/conda-forge/nomkl-feedstock	https://github.com/conda-forge/nomkl-feedstock/blob/main/LICENSE.txt
notebook	http://jupyter.org	https://github.com/jupyter/notebook/blob/main/LICENSE
notebook-shim	https://github.com/jupyter/notebook_shim	https://github.com/jupyter/notebook_shim
notebook_shim	http://jupyter.org	https://github.com/jupyter/notebook_shim/blob/main/LICENSE
numba	https://numba.pydata.org	https://github.com/numba/numba/blob/main/LICENSE
numcpus	https://github.com/tklauser/numcpus	https://github.com/tklauser/numcpus/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
numexpr	https://github.com/pydata/numexpr	https://github.com/pydata/numexpr/blob/master/LICENSE.txt
numpy	https://numpy.org	https://github.com/numpy/numpy/blob/main/LICENSE.txt
numpy-base	https://numpy.org	https://github.com/numpy/numpy/blob/main/LICENSE.txt
numpydoc	https://numpydoc.readthedocs.io/en/latest/index.html	https://github.com/numpy/numpydoc/blob/main/LICENSE.txt
nvidia-cublas-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cublas-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cuda-cupti-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cuda-cupti-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cuda-nvrtc-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cuda-nvrtc-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cuda-runtime-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cuda-runtime-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cudnn-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/deeplearning/cudnn/sla/index.html
nvidia-cudnn-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/deeplearning/cudnn/sla/index.html

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
nvidia-cuEquivariance	https://github.com/NVIDIA/cuEquivariance	https://github.com/NVIDIA/cuEquivariance/blob/main/LICENSES/LICENSE
nvidia-cufft-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cufft-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cufile-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html#
nvidia-curand-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-curand-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cusolver-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cusolver-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cusparse-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cusparse-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-cusparselt-cu12	https://docs.nvidia.com/cuda/cusparselt/index.html	https://docs.nvidia.com/cuda/cusparselt/license.html
nvidia-ml-py	https://pypi.org/project/nvidia-ml-py/	https://pypi.org/project/nvidia-ml-py/
nvidia-nccl-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/hpc-sdk/eula/index.html
nvidia-nccl-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/hpc-sdk/eula/index.html
nvidia-nvjitlink-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
nvidia-nvshmem-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-nvtx-cu11	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
nvidia-nvtx-cu12	https://developer.nvidia.com/cuda-zone	https://docs.nvidia.com/cuda/eula/index.html
Oat++	https://oatpp.io/	https://github.com/oatpp/oatpp/blob/master/LICENSE
oauthlib	https://github.com/oauthlib/oauthlib	https://github.com/oauthlib/oauthlib/blob/master/LICENSE
ocl-icd	https://github.com/OCL-dev/ocl-icd	https://github.com/OCL-dev/ocl-icd/blob/master/COPYING
ocl-icd-system	https://github.com/conda-forge/ocl-icd-system-feedst	https://github.com/conda-forge/ocl-icd-system-feedst blob/main/LICENSE.txt
olefile	https://www.decalage.info/python/olefileio	https://olefile.readthedocs.io/en/latest/License.html
omegaconf	https://github.com/omry/omegaconf	https://github.com/omry/omegaconf/blob/master/LICENSE
OmegaFold	https://github.com/HeliXonProtein/OmegaFold/tree/main	https://github.com/HeliXonProtein/OmegaFold/blob/main/LICENSE
omnicanvas	https://omnicanvas.readthedocs.io/en/latest/	https://github.com/samirelanduk/omnicanvas/blob/master/LICENSE
OpenFF	https://openforcefield.org/	https://github.com/openforcefield/openff-forcefields
openff-amber-ff-ports	https://github.com/openforcefield/openff-amber-ff-ports	https://github.com/openforcefield/openff-amber-ff-ports blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
openff-forcefields	https://openforcefield.org	https://github.com/openforcefield/openff-forcefields/blob/master/LICENSE
openff-fragmenter	https://github.com/openforcefield/openff-fragmenter	https://github.com/openforcefield/openff-fragmenter/blob/main/LICENSE
openff-interchange	https://github.com/openforcefield/openff-interchange	https://github.com/openforcefield/openff-interchange/blob/main/LICENSE
openff-models	https://github.com/openforcefield/openff-models	https://github.com/openforcefield/openff-models/blob/main/LICENSE
openff-nagl	https://docs.openforcefield.org/projects/nagl/en/latest/?badge=latest	https://github.com/openforcefield/openff-nagl/blob/main/LICENSE
openff-nagl-models	https://github.com/openforcefield/openff-nagl-models	https://github.com/openforcefield/openff-nagl-models/blob/main/LICENSE
openff-toolkit	https://openforcefield.org	https://github.com/openforcefield/openff-toolkit/blob/master/LICENSE
openff-toolkit-base	https://openforcefield.org	https://github.com/openforcefield/openff-toolkit/blob/master/LICENSE
openff-units	https://github.com/openforcefield/openff-units	https://github.com/openforcefield/openff-units/blob/main/LICENSE
openff-utilities	https://github.com/openforcefield/openff-utilities	https://github.com/openforcefield/openff-utilities/blob/main/LICENSE
openjpeg	https://github.com/uclouvain/openjpeg	https://github.com/uclouvain/openjpeg/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
openldap	https://www.openldap.org/software/repo.html	https://git.openldap.org/openldap/openldap/-/blob/master/LICENSE
OpenMM	https://openmm.org	https://docs.openmm.org/7.1.0/userguide/library.html#license
openmmtools	https://github.com/choderalab/openmmtools	https://github.com/choderalab/openmmtools/blob/main/LICENSE
openmoltools	https://github.com/choderalab/openmoltools	https://github.com/choderalab/openmoltools/blob/master/LICENSE
openpyxl	https://openpyxl.readthedocs.io/en/stable/	https://foss.heptapod.net/openpyxl/openpyxl/-/blob/branch/3.0/LICENCE.rst
openssl	https://www.openssl.org	https://github.com/openssl/openssl/blob/master/LICENSE.txt
opt-einsum	https://github.com/dgasmith/opt_einsum	https://github.com/dgasmith/opt_einsum/blob/master/LICENSE
opt_einsum	https://github.com/dgasmith/opt_einsum	https://github.com/dgasmith/opt_einsum/blob/main/LICENSE
OptKing	https://github.com/psi-rking/optking	https://github.com/psi-rking/optking/blob/master/LICENSE
optree	https://optree.readthedocs.io/en/latest/	https://github.com/metaopt/optree/blob/main/LICENSE
oscrypto	https://github.com/wbond/oscrypto	https://github.com/wbond/oscrypto/blob/master/LICENSE
overrides	https://github.com/mkorpela/overrides	https://github.com/mkorpela/overrides/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
packaging	https://github.com/pypa/packaging	https://github.com/pypa/packaging/blob/main/LICENSE
packmol	https://leandro.iqm.unicamp.br/m3g/packmol/home.shtml	https://github.com/m3g/packmol/blob/master/LICENSE
packmol_memgen	http://cpclab.uni-duesseldorf.de/	https://gitlab.com/cbjh/memgen/py-memgen/-/blob/main/LICENSE
pandas	https://pandas.pydata.org	https://github.com/pandas-dev/pandas/blob/main/LICENSE
pandocfilters	http://github.com/jgm/pandocfilters	https://github.com/jgm/pandocfilters/blob/master/LICENSE
panedr	https://github.com/MDAnalysis/panedr	https://github.com/MDAnalysis/panedr/blob/master/LICENSE.txt
pango	https://pango.gnome.org	https://gitlab.gnome.org/GNOME/pango/-/blob/main/COPYING
ParmEd	https://parmed.github.io/ParmEd/html/index.html	https://github.com/ParmEd/ParmEd#license
parser	https://github.com/typescript-eslint/typescript-eslint	https://github.com/typescript-eslint/typescript-eslint/raw/master/LICENSE
parso	https://parso.readthedocs.io/en/latest/	https://github.com/davidhalter/parso/blob/master/LICENSE.txt
pathos	https://github.com/uqfoundation/pathos	https://github.com/uqfoundation/pathos/blob/master/LICENSE
patsy	https://patsy.readthedocs.io/en/latest/	https://github.com/pydata/patsy/blob/master/LICENSE.txt

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
pbkdf2	https://golang.org/x/crypto/pbkdf2	https://cs.opensource.google/go/x/crypto/+/master:LICENSE
pbr	https://docs.openstack.org/pbr/latest/	https://opendev.org/openstack/pbr/src/branch/master/LICENSE
pcmsolver	https://pcmsolver.readthedocs.io/en/stable/	https://github.com/PCMSolver/pcmsolver/blob/master/LICENSE
pcre	https://www.pcre.org	https://www.pcre.org/licence.txt
pcre2	https://www.pcre.org	https://github.com/PCRE2Project/pcre2/blob/master/LICENCE
pdb4amber	https://github.com/Amber-MD/pdb4amber	https://github.com/Amber-MD/pdb4amber/blob/master/LICENSE
pdbfixer	https://github.com/openmm/pdbfixer	https://github.com/openmm/pdbfixer/blob/master/LICENSE
pexpect	https://pexpect.readthedocs.io/	https://github.com/pexpect/pexpect/blob/master/LICENSE
pgconn	https://github.com/jackc/pgconn	https://github.com/jackc/pgconn/blob/master/LICENSE
pgio	https://github.com/jackc/pgio	https://github.com/jackc/pgio/blob/master/LICENSE
pgpassfile	https://github.com/jackc/pgpassfile	https://github.com/jackc/pgpassfile/blob/master/LICENSE
pgproto3	https://github.com/jackc/pgproto3/v2	https://github.com/jackc/pgproto3/blob/master/LICENSE
pgtype	https://github.com/jackc/pgtype	https://github.com/jackc/pgtype/blob/master/LICENSE
pgx	https://github.com/jackc/pgx/v4	https://github.com/jackc/pgx/blob/v4.0.1/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
phonopy	https://github.com/phonopy/phono3py	https://github.com/phonopy/phono3py/blob/develop/LICENSE
pickleshare	https://github.com/pickleshare/pickleshare	https://github.com/pickleshare/pickleshare/blob/master/LICENSE
Pillow	https://python-pillow.org	https://github.com/python-pillow/Pillow/blob/main/LICENSE
Pint	https://pint.readthedocs.io/en/stable/	https://github.com/hgrecco/pint/blob/master/LICENSE
pip	https://pip.pypa.io/	https://github.com/pypa/pip/blob/main/LICENSE.txt
pip-licenses	https://github.com/raimon49/pip-licenses	https://github.com/raimon49/pip-licenses/blob/master/LICENSE
pixman	https://pixman.org	https://cgit.freedesktop.org/pixman/tree/COPYING
pkginfo	https://launchpad.net/pkginfo	https://bazaar.launchpad.net/~tseaver/pkginfo/trunk/view/head:/LICENSE.txt
pkgutil-resolve-name	https://github.com/graingert/pkgutil-resolve-name	https://github.com/graingert/pkgutil-resolve-name?tab=License-1-ov-file#readme
pkgutil_resolve_name	https://github.com/graingert/pkgutil-resolve-name	https://github.com/graingert/pkgutil-resolve-name?tab=License-1-ov-file#readme
platformdirs	https://github.com/platformdirs/platformdirs	https://github.com/platformdirs/platformdirs/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
plotly	https://plotly.com/python/	https://github.com/plotly/plotly.py/blob/master/LICENSE.txt
plotly-orca	https://github.com/plotly/orca	https://github.com/plotly/orca/blob/master/LICENSE
plotly.js	https://github.com/plotly/plotly.js	https://github.com/plotly/plotly.js/raw/master/LICENSE
pluggy	https://pluggy.readthedocs.io/en/stable/index.html	https://github.com/pytest-dev/pluggy/blob/main/LICENSE
pooch	https://github.com/fatiando/pooch	https://github.com/fatiando/pooch/blob/main/LICENSE.txt
portalocker	https://github.com/WoLpH/portalocker	https://github.com/wolph/portalocker/blob/develop/LICENSE
pox	https://github.com/uqfoundation/pox	https://github.com/uqfoundation/pox/blob/master/LICENSE
ppft	https://github.com/uqfoundation/ppft	https://github.com/uqfoundation/ppft/blob/master/LICENSE
pq	https://github.com/lib/pq	https://github.com/lib/pq/blob/master/LICENSE.md
ProDy	http://www.csb.pitt.edu/ProDy	https://github.com/prody/ProDy/blob/master/LICENSE.rst
prometheus-client	https://github.com/prometheus/client_python	https://github.com/prometheus/client_python/blob/master/LICENSE
prometheus_client	https://github.com/prometheus/client_python	https://github.com/prometheus/client_python/blob/master/LICENSE
prompt-toolkit	https://python-prompt-toolkit.readthedocs.io/en/stable/	https://github.com/prompt-toolkit/python-prompt-toolkit/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
prompt_toolkit	https://github.com/prompt-toolkit/python-prompt-toolki	https://github.com/prompt-toolkit/python-prompt-toolki tab=BSD-3-Clause-1-ov-f readme
propcache	https://github.com/aio-libs/propcache	https://github.com/aio-libs/propcache?tab=Apache-2.0-1-ov-file#readme
protobuf	https://google.golang.org/protobuf	https://github.com/protocolbuffers/protobuf-go/blob/master/LICENSE
psi4	https://psicode.org	https://github.com/psi4/psi4/blob/master/COPYING.LESSER
psutil	https://psutil.readthedocs.io/en/latest/	https://github.com/giampaolo/psutil/blob/master/LICENSE
psycopg2	https://psycopg.org/	https://github.com/psycopg/psycopg2/blob/master/LICENSE
PTable	https://github.com/kxxoling/PTable	https://github.com/kxxoling/PTable/blob/master/COPYING
pthread-stubs	https://xcb.freedesktop.org	https://github.com/freedesktop/xcb-pthread-stubs/blob/master/COPYING
ptyprocess	https://ptyprocess.readthedocs.io/en/latest/	https://github.com/pexpect/ptyprocess/blob/master/LICENSE
pure-eval	http://github.com/alexmojaki/pure_eval	http://github.com/alexmojaki/pure_eval/blob/master/LICENSE.txt
pure_eval	http://github.com/alexmojaki/pure_eval	https://github.com/alexmojaki/pure_eval?tab=MIT-1-ov-file#readme

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
py	https://py.readthedocs.io/en/latest/	https://github.com/pytest-dev/py/blob/master/LICENSE
py-cpuinfo	https://github.com/workhorsy/py-cpuinfo	https://github.com/workhorsy/py-cpuinfo/blob/master/LICENSE
pyasn1	https://github.com/etingof/pyasn1	https://github.com/etingof/pyasn1/blob/master/LICENSE.rst
pyasn1-modules	https://github.com/etingof/pyasn1-modules	https://github.com/etingof/pyasn1-modules/blob/master/LICENSE.txt
pyasn1_modules	https://pyasn1.readthedocs.io/en/latest/contents.html	https://github.com/pyasn1/pyasn1-modules?tab=BSD-2-Clause-1-ov-file#readme
pybind11	http://pybind11.readthedocs.io/en/stable/	https://github.com/pybind/pybind11/blob/master/LICENSE
pybind11-abi	https://github.com/pybind/pybind11	https://github.com/pybind/pybind11/blob/master/LICENSE
pybind11_global	http://pybind11.readthedocs.io/en/stable/	https://github.com/pybind/pybind11/blob/master/LICENSE
pycairo	https://pycairo.readthedocs.io/en/latest/index.html	https://github.com/pygobject/pycairo/blob/master/COPYING
pycosat	https://github.com/conda/pycosat	https://github.com/conda/pycosat/blob/main/LICENSE
pycparser	https://github.com/eliben/pycparser	https://github.com/eliben/pycparser/blob/master/LICENSE
pydantic	https://pydantic-docs.helpmanual.io	https://github.com/samuelcolvin/pydantic/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
pydantic-core	https://github.com/pydantic/pydantic-core	https://github.com/pydantic/pydantic-core
pyedr	https://github.com/MDAnalysis/panedr	https://github.com/MDAnalysis/panedr/blob/master/LICENSE.txt
pyEMMA	http://github.com/markovmodel/PyEMMA	https://github.com/markovmodel/PyEMMA/blob/devel/LICENSE.txt
Pygments	https://pygments.org	https://github.com/pygments/pygments/blob/master/LICENSE
pygments	https://pygments.org/	https://github.com/pygments/pygments?tab=BSD-2-Clause-1-ov-file#readme
pygraphviz	https://pygraphviz.github.io	https://github.com/pygraphviz/pygraphviz/blob/main/LICENSE
pygtop	https://pygtop.readthedocs.io/en/latest/	https://github.com/samirelanduk/pygtop/blob/master/LICENSE
pyHanko	https://github.com/MatthiasValvekens/pyHanko	https://github.com/MatthiasValvekens/pyHanko/blob/master/LICENSE
pyhanko-certvalidator	https://github.com/MatthiasValvekens/certvalidator	https://github.com/MatthiasValvekens/certvalidator/blob/master/LICENSE
PyJWT	https://github.com/jpadilla/pyjwt	https://github.com/jpadilla/pyjwt/blob/master/LICENSE
pymbar	https://pymbar.org	https://github.com/choderalab/pymbar/blob/master/LICENSE
pynvml	https://github.com/gpuopenanalytics/pynvml	https://github.com/gpuopenanalytics/pynvml/blob/master/LICENSE.txt

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
pyOpenSSL	https://pyopenssl.org/	https://github.com/pyca/pyopenssl/blob/main/LICENSE
pyparsing	https://pyparsing-docs.readthedocs.io/en/latest/	https://github.com/pyparsing/pyparsing/blob/master/LICENSE
PyPDF3	https://github.com/sfneal/PyPDF3	https://github.com/sfneal/PyPDF3/blob/master/LICENSE
pyrsistent	http://github.com/tobgu/pyrsistent/	https://github.com/tobgu/pyrsistent/blob/master/LICENSE.mit
pysam	https://github.com/pysam-developers/pysam	https://github.com/pysam-developers/pysam/blob/master/COPYING
PySocks	https://github.com/Anorov/PySocks	https://github.com/Anorov/PySocks/blob/master/LICENSE
pytables	https://www.pytables.org	https://github.com/PyTables/PyTables/blob/master/LICENSE.txt
python	https://www.python.org/	https://docs.python.org/3/license.html
python-bidi	https://github.com/MeirKriheli/python-bidi	https://github.com/MeirKriheli/python-bidi/blob/master/COPYING
python-constraint	https://github.com/python-constraint/python-constraint	https://github.com/python-constraint/python-constraint/blob/master/LICENSE
python-dateutil	https://dateutil.readthedocs.io	https://github.com/dateutil/dateutil/blob/master/LICENSE
python-json-logger	http://github.com/madzak/python-json-logger	https://github.com/madzak/python-json-logger/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
python-pptx	https://github.com/scanny/python-pptx	https://github.com/scanny/python-pptx/blob/master/LICENSE
python-ldap	https://www.python-ldap.org/	https://github.com/python-ldap/python-ldap/blob/python-ldap-3.4.0/LICENCE
python3-saml	https://github.com/onelogin/python3-saml	https://github.com/onelogin/python3-saml/blob/master/LICENSE
python_abi	https://github.com/conda-forge/python_abi-feedstock	https://github.com/conda-forge/python_abi-feedstock/blob/main/LICENSE.txt
pytorch-lightning	https://github.com/Lightning-AI/pytorch-lightning	https://github.com/Lightning-AI/pytorch-lightning/blob/master/LICENSE
pytz	https://pythonhosted.org/pytz	https://git.launchpad.net/pytz/tree/LICENSE.txt
pytz-deprecation-shim	https://github.com/pganssle/pytz-deprecation-shin	https://github.com/pganssle/pytz-deprecation-shin blob/master/LICENSE
PyWavelets	https://github.com/PyWavelets/pywt	https://github.com/PyWavelets/pywt/blob/master/LICENSE
PyYAML	https://pyyaml.org/	https://github.com/yaml/pyyaml/blob/master/LICENSE
pyyml	No longer available	No longer available
pyzmq	https://pyzmq.readthedocs.io/en/latest/	https://github.com/zeromq/pyzmq/blob/main/COPYING.BSD
qcelemental	https://github.com/MolSSI/QCElemental	https://github.com/MolSSI/QCElemental/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
qcengine	https://github.com/MolSSI/QCEngine	https://github.com/MolSSI/QCEngine/blob/master/LICENSE
qrcode	https://github.com/lincolnloop/python-qrcode	https://github.com/lincolnloop/python-qrcode/blob/main/LICENSE
ramda	https://github.com/ramda/ramda	https://github.com/ramda/ramda/raw/master/LICENSE.txt
rapidjson	https://rapidjson.org/	https://github.com/Tencent/rapidjson/blob/master/license.txt
rdkit	https://www.rdkit.org	https://github.com/rdkit/rdkit/blob/master/license.txt
re2	https://github.com/google/re2	https://github.com/google/re2/blob/main/LICENSE
readme-renderer	https://github.com/pypa/readme_renderer	https://github.com/pypa/readme_renderer/blob/main/LICENSE
redis-py	https://github.com/andymccurdy/redis-py	https://github.com/redis/redis-py/blob/master/LICENSE
referencing	https://github.com/python-jsonschema/referencing	https://github.com/python-jsonschema/referencing
regex	https://github.com/mrabarnett/mrab-regex	https://github.com/mrabarnett/mrab-regex/blob/hg/LICENSE.txt
reportlab	https://www.reportlab.com	https://www.reportlab.com/dev/docs/faq/#1.3
reproc	https://github.com/DaanDeMeyer/reproc	https://github.com/DaanDeMeyer/reproc/blob/main/LICENSE
reproc-cpp	https://github.com/DaanDeMeyer/reproc	https://github.com/DaanDeMeyer/reproc/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
requests	https://requests.readthedocs.io	https://github.com/psf/requests/blob/main/LICENSE
requests-oauthlib	https://github.com/requests/requests-oauthlib	https://github.com/requests/requests-oauthlib/blob/master/LICENSE
requests-toolbelt	https://toolbelt.readthedocs.org	https://github.com/requests/toolbelt/blob/master/LICENSE
resumable	https://github.com/stevvooe/resumable	https://github.com/stevvooe/resumable/blob/master/LICENSE
retrying	https://github.com/rholder/retrying	https://github.com/rholder/retrying/blob/master/LICENSE
rfc3339-validator	https://github.com/naimetti/rfc3339-validator	https://github.com/naimetti/rfc3339-validator/blob/master/LICENSE
rfc3339_validator	https://github.com/naimetti/rfc3339-validator	https://github.com/naimetti/rfc3339-validator?tab=MIT-1-ov-file#readme
rfc3986	https://rfc3986.readthedocs.io/en/latest/	https://github.com/python-hyper/rfc3986/blob/main/LICENSE
rfc3986-validator	https://github.com/naimetti/rfc3986-validator	https://github.com/naimetti/rfc3986-validator/blob/master/LICENSE
rfc3987-syntax	https://github.com/willynilly/rfc3987-syntax	https://github.com/willynilly/rfc3987-syntax?tab=MIT-1-ov-file#readme
rich	https://github.com/Textualize/rich	https://github.com/Textualize/rich/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
rich-argparse	https://github.com/hamdanal/rich-argparse	https://github.com/hamdanal/rich-argparse/blob/main/LICENSE
rlPyCairo	https://pypi.org/project/rlPyCairo/	https://hg.reportlab.com/hg-public/rlPyCairo/file/tip/rlPyCairo/LICENSE.txt
RocksDict	https://github.com/rocksdict/RocksDict	https://github.com/rocksdict/RocksDict?tab=MIT-1-ov-file#readme
roman-numerals	https://github.com/AA-Turner/roman-numerals	https://github.com/AA-Turner/roman-numerals?tab=License-1-ov-file#readme
roman-numerals-py	https://github.com/AA-Turner/roman-numerals	https://github.com/AA-Turner/roman-numerals?tab=License-1-ov-file#readme
rotary-embedding-torch	https://github.com/lucidrains/rotary-embedding-tor	https://github.com/lucidrains/rotary-embedding-tor blob/main/LICENSE
rpds-py	https://github.com/crate-py/rpds	https://github.com/crate-py/rpds
rpmpack	https://github.com/google/rpmpack	https://github.com/google/rpmpack/blob/master/LICENSE
rsa	https://stuvel.eu/rsa	https://github.com/sybrenstuvel/python-rsa/blob/main/LICENSE
ruamel-yaml	https://sourceforge.net/p/ruamel-yaml/code/ci/default/tree/	https://sourceforge.net/p/ruamel-yaml/code/ci/default/tree/LICENSE
ruamel.yaml.clib	https://sourceforge.net/p/ruamel-yaml-clib/code/ci/default/tree/	https://sourceforge.net/p/ruamel-yaml-clib/code/ci/default/tree/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
s3transfer	https://github.com/boto/s3transfer	https://github.com/boto/s3transfer/blob/develop/LICENSE.txt
sacrebleu	https://github.com/mjpost/sacrebleu	https://github.com/mjpost/sacrebleu/blob/master/LICENSE.txt
sapiens	https://github.com/Merck/Sapiens	https://github.com/Merck/Sapiens/blob/main/LICENSE
sasl	https://mellium.im/sasl	https://github.com/mellium/sasl/blob/main/LICENSE
scikit-gstat	https://mmaelicke.github.io/scikit-gstat/	https://github.com/mmaelicke/scikit-gstat/blob/main/LICENSE
scikit-image	https://scikit-image.org	https://scikit-image.org/docs/dev/license.html
scikit-learn	https://scikit-learn.org/stable/	https://github.com/scikit-learn/scikit-learn/blob/main/COPYING
scikit-learn-extra	https://github.com/scikit-learn-contrib/scikit-learn-extra	https://github.com/scikit-learn-contrib/scikit-learn-extra/blob/main/LICENSE
scikit-network	https://github.com/sknetwork-team/scikit-network	https://github.com/sknetwork-team/scikit-network?tab=License-1-ov-file#readme
scipy	https://scipy.org	https://github.com/scipy/scipy/blob/main/LICENSE.txt
seaborn	https://seaborn.pydata.org	https://github.com/mwaskom/seaborn/blob/master/LICENSE
seaborn-base	https://seaborn.pydata.org	https://github.com/mwaskom/seaborn/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
semver	https://github.com/Masterminds/semver/v3	https://github.com/Masterminds/semver/blob/v3.0.3/LICENSE.txt
Send2Trash	https://github.com/arsenetar/send2trash	https://github.com/arsenetar/send2trash/blob/master/LICENSE
sentry-sdk	https://github.com/getsentry/sentry-python	https://github.com/getsentry/sentry-python/blob/master/LICENSE
setproctitle	https://github.com/dvarrazzo/py-setproctitle	https://github.com/dvarrazzo/py-setproctitle/blob/master/LICENSE
setuptools	https://github.com/pypa/setuptools	https://github.com/pypa/setuptools/blob/main/LICENSE
setuptools-scm	https://github.com/pypa/setuptools_scm/	https://github.com/pypa/setuptools_scm/blob/main/LICENSE
sh	https://github.com/amoffat/sh	https://github.com/amoffat/sh/blob/develop/LICENSE.txt
shellingham	https://github.com/sarugaku/shellingham	https://github.com/sarugaku/shellingham/blob/master/LICENSE
simdjson	https://github.com/simdjson/simdjson/tree/master	https://github.com/simdjson/simdjson/blob/master/LICENSE
simint	https://www.bennyp.org/research/simint/	https://github.com/simint-chem/simint-generator/blob/master/LICENSE
simplejson	https://github.com/simplejson/simplejson	https://github.com/simplejson/simplejson/blob/master/LICENSE.txt
six	https://github.com/benjaminp/six	https://github.com/benjaminp/six/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
smirnoff99frosst	https://github.com/openforcefield/smirnoff99frosst	https://github.com/openforcefield/smirnoff99Frosst/blob/master/LICENSE
smmap	https://github.com/gitpython-developers/smmap	https://github.com/gitpython-developers/smmap/blob/master/LICENSE
snappy	https://github.com/google/snappy	https://github.com/google/snappy/blob/main/COPYING
sniffio	https://github.com/python-trio/sniffio	https://github.com/python-trio/sniffio/blob/master/LICENSE.MIT
snowballstemmer	https://github.com/snowballstem/snowball	https://github.com/snowballstem/snowball/blob/master/COPYING
soupsieve	https://github.com/facelessuser/soupsieve	https://github.com/facelessuser/soupsieve/blob/main/LICENSE.md
spdlog	https://github.com/gabime/spdlog	https://github.com/gabime/spdlog?tab=License-1-ov-file#readme
spglib	https://spglib.readthedocs.io/	https://github.com/spglib/spglib/blob/develop/COPYING
sphinx	https://github.com/sphinx-doc/sphinx	https://github.com/sphinx-doc/sphinx/blob/5.x/LICENSE
sphinxcontrib-applehelp	https://github.com/sphinx-doc/sphinxcontrib-applehe	https://github.com/sphinx-doc/sphinxcontrib-applehe blob/master/LICENSE
sphinxcontrib-devhelp	https://github.com/sphinx-doc/sphinxcontrib-devhelp	https://github.com/sphinx-doc/sphinxcontrib-devhelp blob/master/LICENSE
sphinxcontrib-htmlhelp	https://github.com/sphinx-doc/sphinxcontrib-htmlhel	https://github.com/sphinx-doc/sphinxcontrib-htmlhel blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
sphinxcontrib-jsmath	https://github.com/sphinx-doc/sphinxcontrib-jsmath	https://github.com/sphinx-doc/sphinxcontrib-jsmath/blob/master/LICENSE
sphinxcontrib-qthelp	https://github.com/sphinx-doc/sphinxcontrib-qthelp	https://github.com/sphinx-doc/sphinxcontrib-qthelp/blob/master/LICENSE
sphinxcontrib-serializinghtml	https://github.com/sphinx-doc/sphinxcontrib-serializ	https://github.com/sphinx-doc/sphinxcontrib-serializ blob/master/LICENSE
SQLAlchemy	https://www.sqlalchemy.org	https://docs.sqlalchemy.org/en/14/copyright.html
SQLAlchemy	https://www.sqlalchemy.org/	https://github.com/sqlalchemy/sqlalchemy?tab=MIT-1-ov-file#readme
sqlite	https://sqlite.org/index.html	https://sqlite.org/copyright.html
sqlparse	https://github.com/andialbrecht/sqlparse	https://github.com/andialbrecht/sqlparse/blob/master/LICENSE
stack-data	http://github.com/alexmojaki/stack_data	https://github.com/alexmojaki/stack_data/blob/master/LICENSE.txt
stack_data	http://github.com/alexmojaki/stack_data	https://github.com/alexmojaki/stack_data?tab=MIT-1-ov-file#readme
starfile	https://github.com/alisterburt/starfile	https://github.com/teamtomo/starfile/blob/main/LICENSE
statsmodels	https://github.com/statsmodels/statsmodels	https://github.com/statsmodels/statsmodels/blob/main/LICENSE.txt
structlog	https://www.structlog.org/	https://github.com/hynek/structlog/blob/main/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
streamlit	https://github.com/streamlit/streamlit	https://github.com/streamlit/streamlit/blob/develop/LICENSE
svglib	https://github.com/deeplook/svglib	https://github.com/deeplook/svglib/blob/master/LICENSE.txt
sympy	https://sympy.org	https://github.com/sympy/sympy/blob/master/LICENSE
tables	https://www.pytables.org/	https://github.com/PyTables/PyTables/blob/master/LICENSE.txt
tabulate	https://github.com/astanin/python-tabulate	https://github.com/astanin/python-tabulate/blob/master/LICENSE
tbb	https://github.com/oneapi-src/oneTBB	https://github.com/oneapi-src/oneTBB/blob/master/LICENSE.txt
tenacity	https://github.com/jd/tenacity	https://github.com/jd/tenacity/blob/master/LICENSE
tensorboard	https://github.com/tensorflow/tensorboard	https://github.com/tensorflow/tensorboard/blob/master/LICENSE
tensorboard-data-server	https://github.com/tensorflow/tensorboard	https://github.com/tensorflow/tensorboard/blob/master/LICENSE
tensorboard-plugin-wit	https://github.com/pair-code/what-if-tool	https://github.com/PAIR-code/what-if-tool/blob/master/LICENSE
tensorboardX	https://github.com/lanpa/tensorboardX	https://github.com/lanpa/tensorboardX/blob/master/LICENSE
tensorflow	https://github.com/tensorflow/tensorflow	https://github.com/tensorflow/tensorflow/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
tensorflow-estimator	https://github.com/tensorflow/estimator	https://github.com/tensorflow/estimator/blob/master/LICENSE
tensorflow-io-gcs-filesystem	https://github.com/tensorflow/io	https://github.com/tensorflow/io/blob/master/LICENSE
tensorflow-probability	https://github.com/tensorflow/probability	https://github.com/tensorflow/probability/blob/main/LICENSE
termcolor	https://github.com/hugovk/termcolor	https://github.com/hugovk/termcolor/blob/main/COPYING.txt
terminado	https://github.com/jupyter/terminado	https://github.com/jupyter/terminado/blob/main/LICENSE
testpath	https://github.com/jupyter/testpath	https://github.com/jupyter/testpath/blob/master/LICENSE
textangular	https://github.com/fraywing/textAngular	https://github.com/fraywing/textAngular
texttable	https://github.com/foutaise/texttable/	https://github.com/foutaise/texttable/?tab=MIT-1-ov-file#readme
tf_keras	https://github.com/keras-team/keras-io	https://github.com/keras-team/keras-io/blob/master/LICENSE
tf_keras	https://github.com/keras-team/keras-io	https://github.com/keras-team/keras-io/blob/master/LICENSE
threadpoolctl	https://github.com/joblib/threadpoolctl	https://github.com/joblib/threadpoolctl/blob/master/LICENSE
three	https://github.com/mrdoob/three.js	https://github.com/mrdoob/three.js/raw/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
tiffile	https://github.com/cgohlke/tiffile/	https://github.com/cgohlke/tiffile/blob/master/LICENSE
tinycss2	https://github.com/Kozea/tinycss2/	https://github.com/Kozea/tinycss2/blob/master/LICENSE
tinyxml2	https://github.com/leethomason/tinyxml2	https://github.com/leethomason/tinyxml2/blob/master/LICENSE.txt
tk	https://www.tcl.tk/	https://www.tcl.tk/software/tcltk/license.html
toml	https://github.com/toml-lang/toml	https://github.com/toml-lang/toml/blob/main/LICENSE
tomli	https://github.com/hukkin/tomli	https://github.com/hukkin/tomli/blob/master/LICENSE
toolz	https://github.com/pytoolz/toolz	https://github.com/pytoolz/toolz/blob/master/LICENSE.txt
torch	https://pytorch.org/	https://pypi.org/project/torch/#license
torch-geometric	https://pyg.org/	https://github.com/pyg-team/pytorch_geometric?tab=MIT-1-ov-file#readme
torcheval	https://github.com/pytorch/torcheval	https://github.com/pytorch/torcheval?tab=License-1-ov-file#readme
torchmetrics	https://github.com/Lightning-AI/torchmetrics	https://github.com/Lightning-AI/torchmetrics?tab=Apache-2.0-1-ov-file#readme
tornado	https://www.tornadoweb.org	https://github.com/tornadoweb/tornado/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL		
tqdm	https://github.com/tqdm/tqdm	https://github.com/tqdm/tqdm/blob/master/LICENCE		
tracy	https://github.com/gear-genomics/tracy	https://github.com/gear-genomics/tracy/blob/main/LICENSE		
traitlets	https://github.com/ipython/traitlets	https://github.com/ipython/traitlets/blob/main/COPYING.md		
triton	https://github.com/openai/triton/	https://github.com/openai/triton/blob/main/LICENSE		
truststore	https://github.com/sethmlarson/trus	https://github.com/sethmlarson/truststore/blob/main/LICENSE		
truststore	https://github.com/sethmlarson/truststore	https://github.com/sethmlarson/truststore/blob/main/LICENSEtruststore	https://github.com/sethmlarson/truststore	https://github.com/sethmlarson/truststore/blob/main/LICENSE
ts-jest	https://github.com/kulshekhar/ts-jest	https://github.com/kulshekhar/ts-jest/raw/master/LICENSE.md		
ts-loader	https://github.com/TypeStrong/ts-loader	https://github.com/TypeStrong/ts-loader/raw/master/LICENSE		
twine	https://github.com/pypa/twine	https://github.com/pypa/twine/blob/main/LICENSE		
twinj uuid	https://github.com/twinj/uuid	https://github.com/twinj/uuid/blob/master/LICENSE		
types	https://github.com/babel/babel	https://github.com/babel/babel/raw/master/LICENSE		
types-python-dateutil	https://github.com/python/typeshed	https://github.com/python/typeshed?tab=License-1-ov-file#readme		
types-requests	https://pypi.org/project/types-requests/	https://github.com/python/typeshed/blob/main/LICENSE		

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
typescript	https://github.com/Microsoft/TypeScript	https://github.com/Microsoft/TypeScript/raw/master/LICENSE.txt
typing-inspection	https://github.com/pydantic/typing-inspection	https://github.com/pydantic/typing-inspection?tab=MIT-1-ov-file#readme
typing-utils	https://github.com/bojiang/typing_utils	https://github.com/bojiang/typing_utils?tab=Apache-2.0-1-ov-file#readme
typing_extensions	https://github.com/python/typing	https://github.com/python/typing/blob/master/LICENSE
typing_utils	https://github.com/bojiang/typing_utils	https://github.com/bojiang/typing_utils?tab=Apache-2.0-1-ov-file#readme
tzdata	https://github.com/python/tzdata	https://github.com/python/tzdata/blob/master/LICENSE
tzlocal	https://github.com/regebro/tzlocal	https://github.com/regebro/tzlocal/blob/master/LICENSE.txt
umf	https://github.com/oneapi-src/unified-memory-fram	https://github.com/oneapi-src/unified-memory-fram tab=License-1-ov-file#readme
umi-tools	https://github.com/CGATOxford/UMI-tools	https://github.com/CGATOxford/UMI-tools/blob/master/LICENSE
unicodedata2	https://github.com/fonttools/unicodedata2	https://github.com/fonttools/unicodedata2/blob/master/LICENSE
uri-template	https://gitlab.linss.com/open-source/python/uri-template	https://gitlab.linss.com/open-source/python/uri-template/-/blob/main/LICENSE
uritools	https://github.com/tkem/uritools/	https://github.com/tkem/uritools/blob/master/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL		
urllib3	https://urllib3.readthedocs.io/	https://github.com/urllib3/urllib3/blob/main/LICENSE.txt		
vine	https://github.com/celery/vine	https://github.com/celery/vine/blob/master/LICENSE		
vue	https://github.com/vuejs/vue	https://github.com/vuejs/vue/raw/master/LICENSE		
wadler_lindig	https://github.com/patrick-kidger/wadler_lindig	https://github.com/patrick-kidger/wadler_lindig?tab=Apache-2.0-1-ov-file#readme		
wadler_lindig	https://github.com/patrick-kidger/wadler_lindig	https://github.com/patrick-kidger/wadler_lindig?tab=Apache-2.0-1-ov-file#readmewcwidth	https://github.com/jquast/wcwidth	https://github.com/jquast/wcwidth/blob/master/LICENSE
wandb	https://github.com/wandb/wandb	https://github.com/wandb/wandb/blob/main/LICENSE		
webcolors	https://github.com/ubernostrum/webcolors	https://github.com/ubernostrum/webcolors?tab=BSD-3-Clause-1-ov-file#readme		
webencodings	https://github.com/gsnedders/python-webencodings	https://github.com/gsnedders/python-webencodings/blob/master/LICENSE		
websocket-client	https://github.com/websocket-client/websocket-client.git	https://github.com/websocket-client/websocket-client/blob/master/LICENSE		
Werkzeug	https://palletsprojects.com/p/werkzeug/	https://werkzeug.palletsprojects.com/en/2.1.x/license/		
westpa	http://github.com/westpa/westpa	https://github.com/westpa/westpa/blob/westpa2/LICENSE		
wheel	https://github.com/pypa/wheel	https://github.com/pypa/wheel/blob/main/LICENSE.txt		

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
widgetsnbextension	https://github.com/jupyter-widgets/ipywidgets#readme	https://github.com/jupyter-widgets/ipywidgets/blob/main/LICENSE
wrapt	https://github.com/GrahamDumpleton/wrapt	https://github.com/GrahamDumpleton/wrapt/blob/develop/LICENSE
wsutil	https://github.com/yhat/wsutil	https://github.com/yhat/wsutil/blob/master/LICENSE
x/lint	https://golang.org/x/lint	https://github.com/golang/lint/blob/master/LICENSE
x/mod	https://golang.org/x/mod/semver	https://cs.opensource.google/go/x/mod/+/master:LICENSE
x/net	https://golang.org/x/net	https://cs.opensource.google/go/x/net/+/master:LICENSE
x/oauth2	https://golang.org/x/oauth2	https://cs.opensource.google/go/x/oauth2/+/master:LICENSE
x/sys	https://golang.org/x/sys	https://cs.opensource.google/go/x/sys/+/master:LICENSE
x/text	https://golang.org/x/text	https://cs.opensource.google/go/x/text/+/master:LICENSE
x/tools	https://golang.org/x/tools	https://cs.opensource.google/go/x/tools/+/master:LICENSE
x/xerrors	https://golang.org/x/xerrors	https://cs.opensource.google/go/x/xerrors/+/master:LICENSE
xgboost	https://github.com/dmlc/xgboost	https://github.com/dmlc/xgboost/blob/master/LICENSE
xhtml2pdf	http://github.com/xhtml2pdf/xhtml2pdf	https://github.com/xhtml2pdf/xhtml2pdf/blob/master/LICENSE.txt

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
xlrd	https://github.com/python-excel/xlrd	https://github.com/python-excel/xlrd/blob/master/LICENSE
xlsxwriter	https://github.com/jmcnamara/XlsxWriter	https://github.com/jmcnamara/XlsxWriter/blob/main/LICENSE.txt
xmlsec	https://github.com/mehcode/python-xmlsec	https://github.com/mehcode/python-xmlsec/blob/master/LICENSE
xmltodict	https://github.com/martinblech/xmltodict	https://github.com/martinblech/xmltodict/blob/master/LICENSE
xorg-kbproto	https://gitlab.freedesktop.org/xorg/proto/kbproto	https://gitlab.freedesktop.org/xorg/proto/kbproto/-/blob/master/COPYING
xorg-libice	https://gitlab.freedesktop.org/xorg/lib/libice	https://gitlab.freedesktop.org/xorg/lib/libice/-/blob/master/COPYING
xorg-libsm	https://gitlab.freedesktop.org/xorg/lib/libsm	https://gitlab.freedesktop.org/xorg/lib/libsm/-/blob/master/COPYING
xorg-libx11	https://gitlab.freedesktop.org/xorg/lib/libx11	https://gitlab.freedesktop.org/xorg/lib/libx11/-/blob/master/COPYING
xorg-libxau	https://gitlab.freedesktop.org/xorg/lib/libxau	https://gitlab.freedesktop.org/xorg/lib/libxau/-/blob/master/COPYING
xorg-libxdmcp	https://gitlab.freedesktop.org/xorg/lib/libxdmcp	https://gitlab.freedesktop.org/xorg/lib/libxdmcp/-/blob/master/COPYING

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
xorg-libxext	https://gitlab.freedesktop.org/xorg/lib/libxext	https://gitlab.freedesktop.org/xorg/lib/libxext/-/blob/master/COPYING
xorg-libxrender	https://gitlab.freedesktop.org/xorg/lib/libxrender	https://gitlab.freedesktop.org/xorg/lib/libxrender/-/blob/master/COPYING
xorg-libxt	https://gitlab.freedesktop.org/xorg/lib/libxt	https://gitlab.freedesktop.org/xorg/lib/libxt/-/blob/master/COPYING
xorg-renderproto	https://gitlab.freedesktop.org/xorg/proto/renderproto	https://gitlab.freedesktop.org/xorg/proto/renderproto/-/blob/master/COPYING
xorg-xextproto	https://gitlab.freedesktop.org/xorg/proto/xextproto	https://gitlab.freedesktop.org/xorg/proto/xextproto/-/blob/master/COPYING
xorg-xproto	https://gitlab.freedesktop.org/xorg/proto/xproto	https://gitlab.freedesktop.org/xorg/proto/xproto/-/blob/master/COPYING
xxhash	https://github.com/cespare/xxhash/v2	https://github.com/cespare/xxhash/blob/main/LICENSE.txt
xz	https://github.com/ulikunitz/xz	https://github.com/ulikunitz/xz/blob/master/LICENSE
yaml	https://pyyaml.org/	https://github.com/yaml/pyyaml/blob/master/LICENSE
yaml-cpp	https://github.com/jbeder/yaml-cpp	https://github.com/jbeder/yaml-cpp/blob/master/LICENSE
yaml.v2	https://gopkg.in/yaml.v2	https://github.com/go-yaml/yaml/blob/v2.4.0/LICENSE
yaml.v3	https://gopkg.in/yaml.v3	https://github.com/go-yaml/yaml/blob/v3/LICENSE

continues on next page

Table 1 – continued from previous page

Name of Project	Website	License URL
yarl	https://github.com/aio-libs/yarl/	https://github.com/aio-libs/yarl/blob/master/LICENSE
yaspin	https://github.com/pavdmyt/yaspin	https://github.com/pavdmyt/yaspin/blob/master/LICENSE
yfiles	https://www.yworks.com/products/yfiles	commercial license
yml	https://pypi.org/project/yml/	N/A
zap	https://go.uber.org/zap	https://github.com/uber-go/zap/blob/master/LICENSE.txt
zipp	https://github.com/jaraco/zipp	https://github.com/jaraco/zipp/blob/main/LICENSE
zlib	https://zlib.net/	https://zlib.net/zlib_license.html
zstandard	https://github.com/indygreg/python-zstandard	https://github.com/indygreg/python-zstandard/blob/main/LICENSE
zstd	https://facebook.github.io/zstd/	https://github.com/facebook/zstd/blob/dev/LICENSE
click-option-group	https://github.com/click-contrib/click-option-group	https://github.com/click-contrib/click-option-group/blob/master/LICENSE

23.1 GCC

On the Linux platform, The OpenEye toolkits are linked with the GCC libraries using an eligible compilation process under the terms of the GCC RUNTIME LIBRARY EXCEPTION (see below).

The OpenEye toolkits are NOT open source software and are NOT governed by GPL or other open source licenses. Your right to use the toolkits are governed by the OpenEye license which is binding to You.

For certain Linux distributions, GCC library components are included as object files. Your rights to use GCC library components (libstdc++, libgcc, and libgomp) included with these distributions are governed by the GPLv3 license as follows:

The components libstdc++, libgcc, and libgomp are part of GCC.

- GCC is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
- GCC is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the

implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with GCC (see below). If not, see Licenses GNU Project .

Per GPLv3, section 6d, the source for GCC is available for download from the following location: GNU. The GCC library components included here are from version gcc-4.9.3 and are built using the standard instructions at Installing GCC.

23.1.1 GCC RUNTIME LIBRARY EXCEPTION

```
Version 3.1, 31 March 2009

Copyright © 2009 Free Software Foundation, Inc. http://fsf.org/

Everyone is permitted to copy and distribute verbatim copies of this license document,␣
↪but changing it is not allowed.

This GCC Runtime Library Exception ("Exception") is an additional permission under␣
↪section 7 of the GNU General Public License,
version 3 ("GPLv3"). It applies to a given file (the "Runtime Library") that bears a␣
↪notice placed by the copyright holder
of the file stating that the file is governed by GPLv3 along with this Exception.

When you use GCC to compile a program, GCC may combine portions of certain GCC header␣
↪files and runtime libraries with
the compiled program. The purpose of this Exception is to allow compilation of non-GPL␣
↪(including proprietary) programs
to use, in this way, the header files and runtime libraries covered by this Exception.

0. Definitions.

A file is an "Independent Module" if it either requires the Runtime Library for␣
↪execution after a Compilation Process,
or makes use of an interface provided by the Runtime Library, but is not otherwise based␣
↪on the Runtime Library.

"GCC" means a version of the GNU Compiler Collection, with or without modifications,␣
↪governed by version 3 (or a specified
later version) of the GNU General Public License (GPL) with the option of using any␣
↪subsequent versions published by the FSF.

"GPL-compatible Software" is software whose conditions of propagation, modification and␣
↪use would permit combination with
GCC in accord with the license of GCC.

"Target Code" refers to output from any compiler for a real or virtual target processor␣
↪architecture, in executable
form or suitable for input to an assembler, loader, linker and/or execution phase.␣
↪Notwithstanding that, Target Code
does not include data in any format that is used as a compiler intermediate␣
↪representation, or used for producing a
compiler intermediate representation.
```

(continues on next page)

(continued from previous page)

```
The "Compilation Process" transforms code entirely represented in non-intermediate␣
↪languages designed for human-written
code, and/or in Java Virtual Machine byte code, into Target Code. Thus, for example, use␣
↪of source code generators and
preprocessors need not be considered part of the Compilation Process, since the␣
↪Compilation Process can be understood
as starting with the output of the generators or preprocessors.

A Compilation Process is "Eligible" if it is done using GCC, alone or with other GPL-
↪compatible software, or if it is
done without using any work based on GCC. For example, using non-GPL-compatible Software␣
↪to optimize any GCC intermediate
representations would not qualify as an Eligible Compilation Process.

1. Grant of Additional Permission.

You have permission to propagate a work of Target Code formed by combining the Runtime␣
↪Library with Independent Modules,
even if such propagation would otherwise violate the terms of GPLv3, provided that all␣
↪Target Code was generated by
Eligible Compilation Processes. You may then convey such a combination under terms of␣
↪your choice, consistent with the
licensing of the Independent Modules.

2. No Weakening of GCC Copyleft.

The availability of this Exception does not imply any general presumption that third-
↪party software is unaffected by
the copyleft requirements of the license of GCC.
```

See also

- http://www.gnu.org/licenses/gcc-exception.html

23.1.2 GNU GENERAL PUBLIC LICENSE

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

 The GNU General Public License is a free, copyleft license for
 software and other kinds of works.

   The licenses for most software and other practical works are designed
```

(continues on next page)

(continued from previous page)

```
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
```

(continues on next page)

(continued from previous page)

```
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                    TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.
```

(continues on next page)

(continued from previous page)

```
  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
```

(continues on next page)

(continued from previous page)

```
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.


  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

   a) The work must carry prominent notices stating that you modified
   it, and giving a relevant date.

   b) The work must carry prominent notices stating that it is
   released under this License and any conditions added under section
   7.  This requirement modifies the requirement in section 4 to
   "keep intact all notices".
```

(continues on next page)

(continued from previous page)

```
    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

  a) Convey the object code in, or embodied in, a physical product
  (including a physical distribution medium), accompanied by the
  Corresponding Source fixed on a durable physical medium
  customarily used for software interchange.

  b) Convey the object code in, or embodied in, a physical product
  (including a physical distribution medium), accompanied by a
  written offer, valid for at least three years and valid for as
  long as you offer spare parts or customer support for that product
  model, to give anyone who possesses the object code either (1) a
  copy of the Corresponding Source for all the software in the
  product that is covered by this License, on a durable physical
  medium customarily used for software interchange, for a price no
  more than your reasonable cost of physically performing this
  conveying of source, or (2) access to copy the
  Corresponding Source from a network server at no charge.

  c) Convey individual copies of the object code with a copy of the
  written offer to provide the Corresponding Source.  This
  alternative is allowed only occasionally and noncommercially, and
  only if you received the object code with such an offer, in accord
```

(continues on next page)

(continued from previous page)

```
   with subsection 6b.

   d) Convey the object code by offering access from a designated
   place (gratis or for a charge), and offer equivalent access to the
   Corresponding Source in the same way through the same place at no
   further charge.  You need not require recipients to copy the
   Corresponding Source along with the object code.  If the place to
   copy the object code is a network server, the Corresponding Source
   may be on a different server (operated by you or a third party)
   that supports equivalent copying facilities, provided you maintain
   clear directions next to the object code saying where to find the
   Corresponding Source.  Regardless of what server hosts the
   Corresponding Source, you remain obligated to ensure that it is
   available for as long as needed to satisfy these requirements.

   e) Convey the object code using peer-to-peer transmission, provided
   you inform other peers where the object code and Corresponding
   Source of the work are being offered to the general public at no
   charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
```

(continues on next page)

(continued from previous page)

```
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
```

(continues on next page)

(continued from previous page)

```
   authors of the material; or

   e) Declining to grant rights under trademark law for use of some
   trade names, trademarks, or service marks; or

   f) Requiring indemnification of licensors and authors of that
   material by anyone who conveys the material (or modified versions of
   it) with contractual assumptions of liability to the recipient, for
   any liability that these contractual assumptions directly impose on
   those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.
```

(continues on next page)

(continued from previous page)

```
  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
```

(continues on next page)

(continued from previous page)

```
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
```

(continues on next page)

(continued from previous page)

```
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
```

(continues on next page)

(continued from previous page)

```
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>
```

(continues on next page)

(continued from previous page)

```
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

See also

- http://www.gnu.org/licenses/gpl.txt

INDEX

A

About, 177
Active, 191
ActiveConformer, 191
ActiveID, 191
ActiveKey, 191
Add, 300
AddCSVSmiles, 300
AddURLMol, 300
AnnotationGet, 241
AnnotationHas, 241
AnnotationSet, 242
AppAddCallback, 177
AppComponentNameGet, 202
AppComponentNameSet, 202
AppCurrentDir, 177
AppCurrentDirSet, 178
AppDir, 178
AppDocDir, 178
AppExampleDir, 178
AppGeometryGet, 203
AppGeometrySet, 203
AppGlobalFontGet, 203
AppGlobalFontSet, 203
AppInstallDir, 178
AppLastDirGet, 178
AppLastDirSet, 178
AppLastOpenDirGet, 178
AppLastOpenDirSet, 179
AppLastSaveDirGet, 179
AppLastSaveDirSet, 179
AppLayoutGet, 203
AppLayoutSet, 203
AppLayoutToIcon, 203
AppLicenseFile, 179
AppLicenseUpdate, 179
AppMainWindowGet, 204
AppMainWindowSet, 204
AppMovableWindowsGet, 204
AppMovableWindowsSet, 204
AppOpenUrl, 179
AppPerspectiveSet, 204
AppRecordWindowEventsGet, 204
AppRecordWindowEventsSet, 204
AppRemoveCallback, 179
AppScriptSave, 179
AppShowGet, 204
AppShowMenuBarGet, 205
AppShowMenuBarSet, 205
AppShowSet, 205
AppShowStatusBarGet, 205
AppShowStatusBarSet, 205
AppSloppyFocusGet, 205
AppSloppyFocusSet, 205
AppStatusTextSet, 206
AppUserDir, 180
AppVersion, 180
AppVersionBugFix, 180
AppVersionMajor, 180
AppVersionMinor, 180
AppWindowStyleGet, 206
AppWindowStyleSet, 206
AtomAddHydrogens, 325
AtomAddHydrogensScoped, 325
AtomAtomicNumDefaultGet, 325
AtomAtomicNumDefaultSet, 325
AtomAtomicNumGet, 326
AtomAtomicNumSet, 326
AtomAtomicNumSetScoped, 326
AtomAttach, 326
AtomClearColorScoped, 242
AtomColorPaletteUpdate, 242
AtomColorReferenceScoped, 242
AtomColorResidueScoped, 242
AtomColorSetScoped, 243
AtomDarkColorsGet, 243
AtomDarkColorsSet, 243
AtomDataGet, 300
AtomDefaultColorGet, 243
AtomDefaultColorSet, 244
AtomDelete, 326
AtomDeleteHydrogens, 326
AtomDeleteHydrogensScoped, 326
AtomDeleteScoped, 327

AtomFormalChargeDefaultGet, 327
AtomFormalChargeDefaultSet, 327
AtomFormalChargeGet, 327
AtomFormalChargeModify, 327
AtomFormalChargeModifyScoped, 327
AtomFormalChargeSet, 327
AtomFormalChargeSetScoped, 327
AtomFuse, 328
AtomHaloRadiusGet, 244
AtomHaloRadiusSet, 244
AtomHybridizationGet, 328
AtomHybridizationSet, 328
AtomHybridizationSetScoped, 328
AtomHydrogenStyleGet, 244
AtomHydrogenStyleSet, 244
AtomHydrogenStyleSetScoped, 244
AtomIsotopeGet, 328
AtomIsotopeSet, 328
AtomIsotopeSetScoped, 328
AtomLabelDefaultGet, 244
AtomLabelDefaultSet, 245
AtomLabelGet, 245
AtomLabelSet, 245
AtomLabelSetScoped, 245
AtomSprout, 329
AtomSproutScoped, 329
AtomStereoDefaultGet, 329
AtomStereoDefaultSet, 329
AtomStereoSet, 329
AtomStereoSetScoped, 329
AtomStereoToggle, 330
AtomStereoToggleScoped, 330
AtomStyleGet, 246
AtomStyleGetScoped, 246
AtomStyleLargeGet, 246
AtomStyleLargeSet, 246
AtomStyleNucleicGet, 246
AtomStyleNucleicSet, 246
AtomStyleProteinGet, 247
AtomStyleProteinSet, 247
AtomStyleSet, 247
AtomStyleSetScoped, 247
AtomStyleToEnum, 248
AtomStyleToText, 248

B

BlockBegin, 206
BlockEnd, 206
BlockExemptGet, 206
BlockExemptSet, 206
BlockGet, 207
BondAngleGet, 330
BondAngleModify, 330
BondAngleSet, 330
BondBallToStickRatioGet, 248
BondBallToStickRatioSet, 248
BondClearColorScoped, 248
BondColorSetScoped, 248
BondCreate, 330
BondDelete, 331
BondDeleteScoped, 331
BondDrawAromaticGet, 248
BondDrawAromaticSet, 248
BondFuse, 331
BondHideHydrogenGet, 249
BondHideHydrogenSet, 249
BondHideNonbondedGet, 249
BondHideNonbondedSet, 249
BondLabelDefaultGet, 249
BondLabelDefaultSet, 249
BondLabelGet, 249
BondLabelSet, 249
BondLabelSetScoped, 250
BondLengthGet, 331
BondLengthModify, 331
BondLengthSet, 332
BondLengthSetScoped, 332
BondLineWidthGet, 250
BondLineWidthSet, 250
BondOrderDefaultGet, 332
BondOrderDefaultSet, 332
BondOrderGet, 332
BondOrderSet, 332
BondOrderSetScoped, 333
BondRadiusGet, 250
BondRadiusSet, 250
BondShowAromaticGet, 250
BondShowAromaticSet, 251
BondShowDipolarGet, 251
BondShowDipolarSet, 251
BondShowOrdersGet, 251
BondShowOrdersSet, 251
BondStereoDefaultGet, 333
BondStereoDefaultSet, 333
BondStereoSet, 333
BondStereoSetScoped, 333
BondStereoToggle, 334
BondStereoToggleScoped, 334
BondStyleGet, 251
BondStyleGetScoped, 251
BondStyleLargeGet, 251
BondStyleLargeSet, 252
BondStyleNucleicGet, 252
BondStyleNucleicSet, 252
BondStyleProteinGet, 252
BondStyleProteinSet, 252
BondStyleSet, 253
BondStyleSetScoped, 253

BondStyleToEnum, 253
BondStyleToText, 253
BondTorsionGet, 334
BondTorsionModify, 334
BondTorsionSet, 334
BookmarkDelete, 253
BookmarkExists, 253
BookmarkLoad, 254
BookmarkMoveDown, 254
BookmarkMoveUp, 254
BookmarkOrganizeDialog, 254
BookmarkRename, 254
Bookmarks, 254
BookmarkSave, 254
BookmarksClear, 255
BookmarksLastLoaded, 255
BuilderActiveGet, 335
BuilderActiveSet, 335
BuilderAlternateConfModeGet, 335
BuilderAlternateConfModeSet, 335
BuilderAlternateConfSet, 335
BuilderCancel, 335
BuilderEdit, 335
BuilderFinish, 335
BuilderFocusOnProperties, 207
BuilderFocusOnSketcher, 207
BuilderFragmentGet, 336
BuilderFragmentSet, 336
BuilderMinimize, 336
BuilderMinimizeScoped, 336
BuilderPropertiesGet, 336
BuilderPropertyAdd, 336
BuilderPropertyRemove, 337
BuilderPropertySetValue, 337
BuilderTorsionActivate, 337
BuilderTorsionsDeactivate, 337
BuilderUseMMFF94sGet, 337
BuilderUseMMFF94sSet, 337

C

CATraceRadiusGet, 255
CATraceRadiusSet, 255
CATraceStyleGet, 255
CATraceStyleSet, 255
CATraceStyleSetScoped, 255
CenterSet, 255
CenterSetScoped, 256
CheckIn, 300
ChildrenGet, 300
ClearActive, 192
ClearLocked, 192
ClearMarked, 192
ClearSelection, 192
ClearVisible, 192
ColorGet, 256
ColorSet, 256
ColorSetScoped, 256
ColorUniqueScoped, 256
ContextClear, 192
ContextGet, 192
ContextInsert, 192
ContextKeysSet, 193
ContextSet, 193
ContourAutoCenterGet, 256
ContourAutoCenterSet, 256
ContourAutoContourGet, 256
ContourAutoContourRadiusScaleSet, 257
ContourAutoContourSet, 257
ContourCenter, 257
ContourCloudJitterDefaultGet, 301
ContourCloudJitterGet, 301
ContourCloudJitterSet, 301
ContourColorForIndexGet, 257
ContourColorForIndexGetScoped, 257
ContourColorForIndexSet, 257
ContourColorForIndexSetScoped, 257
ContourColorSet, 257
ContourColorSetScoped, 258
ContourCountScoped, 301
ContourCreate, 301
ContourCreateSurface, 301
ContourCurrentGridGet, 355
ContourCurrentGridSet, 355
ContourDeleteAll, 301
ContourDeleteByIndex, 302
ContourDeleteByThreshold, 302
ContourDrawAsSurfaceGet, 258
ContourDrawAsSurfaceSet, 258
ContourDrawAsSurfaceSetScoped, 258
ContourDrawStyleGet, 258
ContourDrawStyleSet, 258
ContourDrawStyleSetScoped, 258
ContourEntireGridGet, 258
ContourEntireGridSet, 259
ContourExtractIsoSurface, 302
ContourFixAsSurfaceScoped, 302
ContourGetAll, 302
ContourHideIndex, 259
ContourHideIndexGet, 259
ContourHideIndexSet, 259
ContourLevelForIndexGet, 302
ContourLevelForIndexGetScoped, 302
ContourLevelForIndexSet, 302
ContourLevelForIndexSetScoped, 303
ContourLevelNudgeScoped, 303
ContourLevelSetScoped, 303
ContourLineWidthGet, 259
ContourLineWidthSet, 259

ContourMax, 303
ContourMaxScoped, 303
ContourMin, 303
ContourMinScoped, 303
ContourPickIsoSurfacesGet, 259
ContourPickIsoSurfacesSet, 259
ContourRadiusGet, 303
ContourRadiusSet, 304
ContourResolutionGet, 304
ContourResolutionSet, 304
ContourTransparencySet, 260
ContourTypedAddScoped, 304
ContourTypedCountScoped, 304
ContourTypedMaxScoped, 304
ContourTypedMinScoped, 304
ContourTypedRemoveScoped, 304
ContourTypedSetLevelForIndexScoped, 305
ContourVolume, 305
CopyData, 180
CopyMolecules, 180
CopyMoleculesScoped, 180
CreateAngleMonitor, 356
CreateDistanceMonitor, 356
CreateSphereMonitor, 356
CreateTorsionMonitor, 356
CustomViewDocking, 356
CustomViewEON, 260
CustomViewFRED, 260
CustomViewROCS, 260

D

DataAdd, 340
DataGetDB, 340
DataGetTable, 340
DatatableAddColumn, 340
DatatableCommitChanges, 340
DatatableCurrent, 356
DatatableCurrentGet, 340
DatatableCurrentSet, 340
DatatableData, 340
DatatableDeleteColumn, 341
DatatableEditableGet, 341
DatatableEditableSet, 341
DatatableFilter, 341
DatatableFromList, 341
DatatableGetColumn, 342
DatatableGetCurrentRow, 342
DatatableGetDatatables, 342
DatatableGetImageStreamAtRow, 342
DatatableGetKeys, 342
DatatableGetNumRows, 342
DatatableHeaders, 342
DatatableLingoSimSort, 343
DatatableMolNumberFunction, 343
DatatableMolStringFunction, 343
DatatableNumRows, 344
DatatableSetData, 344
DatatableSetExpression, 344
DatatableSetRowData, 344
DefaultColorLabelGet, 260
DefaultColorLabelSet, 261
DefaultColorMarkedGet, 261
DefaultColorMarkedSet, 261
DefaultColorReferenceGet, 261
DefaultColorReferenceSet, 261
DefaultColorSelectedGet, 261
DefaultColorSelectedSet, 261
DefaultColorTitleGet, 261
DefaultColorTitleSet, 261
DefaultMonitorColorGet, 262
DefaultMonitorColorSet, 262
DefaultScopeGet, 193
DefaultScopeSet, 193
Delete, 305
DeleteAll, 305
DeleteAngleMonitor, 357
DeleteDistanceMonitor, 357
DeleteScoped, 305
DeleteTorsionMonitor, 357
DeleteVisibleMonitors, 357
DistanceControlsVisibilityGet, 262
DistanceControlsVisibilitySet, 262
DrawAtomsAndBondsGet, 262
DrawAtomsAndBondsSet, 262
DrawAxesGet, 262
DrawAxesSet, 262
DrawCATracesGet, 263
DrawCATracesSet, 263
DrawCATracesSetScoped, 263
DrawContoursGet, 263
DrawContoursSet, 263
DrawLabelsGet, 263
DrawLabelsSet, 263
DrawMatrixGet, 263
DrawMatrixSet, 264
DrawRibbonsGet, 264
DrawRibbonsSet, 264
DrawRibbonsSetScoped, 264
DrawSurfacesGet, 264
DrawSurfacesSet, 264
DrawSymmetryGet, 264
DrawSymmetrySet, 264
DrawTitlesGet, 265
DrawTitlesSet, 265
DrawUnitCellGet, 265
DrawUnitCellSet, 265

E

Error, 181
ErrorDetailsDialog, 181
ExistsAngleMonitor, 357
ExistsDistanceMonitor, 357
ExistsTorsionMonitor, 357
ExtensionsEdit, 181

F

FindByDataScoped, 305
FindByQueryScoped, 305
FindBySimilarityScoped, 306
FindBySMARTSScoped, 306
FindByTitleScoped, 306
FindInRepository, 306
FindOnDisk, 306

G

GetAllContextKeys, 193
GetAtomsByScope, 358
GetAtomsScoped, 194
GetBondsByScope, 358
GetBondsScoped, 194
GetContoursByScope, 358
GetContoursScoped, 194
GetGridsByScope, 358
GetGridsScoped, 194
GetKey, 358
GetKeysByScope, 358
GetKeysForID, 358
GetKeyType, 358
GetMarkedAtoms, 194
GetMarkedAtomsScoped, 194
GetMarkedBonds, 194
GetMarkedBondsScoped, 194
GetMarkedContours, 194
GetMarkedGrids, 195
GetMarkedMolecules, 195
GetMarkedMonitors, 195
GetMarkedSurfaces, 195
GetMoleculesByScope, 359
GetMoleculesScoped, 195
GetName, 359
GetPropertyType, 359
GetScoped, 195
GetSelectedAtom, 195
GetSelectedAtoms, 195
GetSelectedBond, 196
GetSelectedBonds, 196
GetSelectedContours, 196
GetSelectedGrids, 196
GetSelectedMolecules, 196
GetSelectedMonitors, 196
GetSelectedSurfaces, 196
GetSurfacesByScope, 359
GetSurfacesScoped, 196
GetVisibleAtoms, 197
GetVisibleBonds, 197
GetVisibleContours, 197
GetVisibleGrids, 197
GetVisibleIDs, 197
GetVisibleMolecules, 197
GetVisibleMonitors, 197
GetVisibleSurfaces, 197
GotoStylePage, 359
GridAdd, 306
GridCheckIn, 307
GridCheckOut, 307
GridClear, 307
GridCopy, 307
GridCreateElectrostaticsGrid, 308
GridCreateGaussian, 308
GridCreateGaussianProduct, 308
GridDefaultContourColorByIndexGet, 265
GridDefaultContourColorByIndexSet, 265
GridDefaultContourLevelByIndexGet, 308
GridDefaultContourLevelByIndexSet, 308
GridDefaultDrawAsSurfaceGet, 265
GridDefaultDrawAsSurfaceSet, 265
GridDefaultNumContoursGet, 308
GridDefaultNumContoursSet, 308
GridInitializeContours, 309
GridNormalize, 309
GridRegularize, 309
GridShowCornersGet, 266
GridShowCornersSet, 266
GridShowLastMaskedGrid, 266
GridToGaussianGrid, 309
GridTypeGet, 309
GridTypeGetScoped, 309
GridTypeSet, 309
GridTypeSetScoped, 309
GridWorkingGetScoped, 309

H

HaloColorDefaultGet, 267
HaloColorDefaultSet, 268
HaloColorGet, 268
HaloColorSet, 268
HaloColorSetScoped, 268
HaloRadiusGet, 268
HaloRadiusSet, 268
HaloRadiusSetScoped, 268
HaloScaleGet, 268
HaloScaleSet, 269
HaloScaleSetScoped, 269
HasGridChildrenScoped, 310

HasSurfaceChildrenScoped, 310
HBondAddTarget, 266
HBondAddTargetsScoped, 266
HBondClearTargets, 266
HBondColorGet, 266
HBondColorSet, 267
HBondRemoveTarget, 267
HBondRemoveTargetsScoped, 267
HBondShowExternalGet, 267
HBondShowExternalSet, 267
HBondShowInternalGet, 267
HBondShowInternalSet, 267
HideNoneScoped, 269
HideOthers, 269
HideScoped, 269

I

IconGetXPM, 207
IDGet, 197
IDTypeGet, 198
Initialize, 310
InitializeObject, 359
InterpreterClear, 207
InterpreterInteractiveGet, 359
InterpreterInteractiveSet, 359
InterpreterPopUpdateGUI, 207
InterpreterPushUpdateGUI, 207
InterpreterShowDebugTextGet, 207
InterpreterShowDebugTextSet, 208
InterpreterTimerGet, 208
InterpreterTimerSet, 208
InterpreterUpdatesGUI, 208
IsActive, 198
IsAGrid, 310
IsAList, 310
IsAMolecule, 310
IsAReflection, 310
IsASmallMolecule, 311
IsASurface, 311
IsLicensed, 181
IsLocked, 198
IsMarked, 198
IsSelected, 198
IsTrulyVisible, 198
IsVisible, 198

J

JournalInit, 181

K

KeyGet, 311
KeyIDGet, 311
KeyParentIDGet, 311
KeysGet, 312
KeySourceIDGet, 311
KeyTypeGet, 311

L

LabelClearColorScoped, 269
LabelClearScoped, 269
LabelColorScoped, 269
LabelDefaultColorGet, 270
LabelDefaultColorSet, 270
LabelFixedSizeGet, 270
LabelFixedSizeSet, 270
LabelGet, 270
LayoutCurrentGet, 208
LayoutExists, 208
LayoutGet, 208
LayoutIcon, 208
LayoutLoad, 209
LayoutOrganizePrompt, 209
LayoutOverrideOrder, 209
LayoutRemove, 209
LayoutRename, 209
Layouts, 210
LayoutSaveCurrent, 209
LayoutSet, 209
LayoutStickyGet, 209
LayoutStickySet, 210
ListAddObject, 312
ListAddObjects, 312
ListGetNames, 312
ListGetObjectLists, 312
ListGetObjects, 312
ListMoveObject, 312
ListMoveObjects, 313
ListNew, 313
ListNewAnd, 313
ListNewMarked, 313
ListNewOr, 313
ListNewXor, 313
ListRemoveObject, 314
ListRemoveObjects, 314
ListRootList, 314
ListSubsetMarked, 314
ListSubsetQuery, 314
ListWindowCollapseAll, 210
ListWindowCollapseCurrent, 210
ListWindowExpandAll, 210
ListWindowExpandCurrent, 210
ListWindowFirstList, 210
ListWindowFirstListItem, 210
ListWindowFocusOnOERID, 210
ListWindowGetCurrentPos, 211
ListWindowHideColumn, 211
ListWindowIsVisible, 211
ListWindowLastList, 211

ListWindowLastListItem, 211
ListWindowNavigateChildrenGet, 211
ListWindowNavigateChildrenSet, 211
ListWindowNavigateLeft, 211
ListWindowNavigateRight, 211
ListWindowNextList, 212
ListWindowNextListItem, 212
ListWindowPrevList, 212
ListWindowPrevListItem, 212
ListWindowRowColoringGet, 212
ListWindowRowColoringSet, 212
ListWindowSetCurrentPos, 212
ListWindowShowColumn, 212
Lock, 198
LockScoped, 199

M

MainWindowScreenshot, 212
MainWindowScreenshotPrompt, 213
Mark, 199
MarkObjectsByScope, 359
MarkScoped, 199
MarkState, 360
MenuAddButton, 213
MenuAddLabel, 360
MenuAddRadioButton, 213
MenuAddSeparator, 213
MenuAddSubmenu, 213
MenuAddToggleButton, 214
MenuBeginRadioGroup, 214
MenuButtonActionSet, 214
MenuDisplayName, 214
MenuDynamicSet, 215
MenuEnableItem, 215
MenuEnableItemCommand, 215
MenuEndRadioGroup, 215
MenuExists, 215
MenuGetAllItems, 215
MenuGetAllItemsContainingName, 215
MenuHasItem, 216
MenuItemIsAMenu, 216
MenuRemoveAll, 216
MenuRemoveItem, 216
MenuUpdateItem, 216
MenuUseTearoffsGet, 360
MenuUseTearoffsSet, 360
MimicParentVisibilityGet, 199
MimicParentVisibilitySet, 199
MimicParentVisibilitySetScoped, 199
MoleculeAdd, 314
MoleculeAddExplicitHydrogens, 338
MoleculeAddHydrogens, 338
MoleculeAddHydrogensScoped, 338
MoleculeAltLocationShow, 270
MoleculeAltLocationVisible, 270
MoleculeAtomBondStyleSetScoped, 271
MoleculeCheckIn, 315
MoleculeCheckOut, 315
MoleculeColorByScoped, 271
MoleculeColorsResetScoped, 271
MoleculeComponentNamesGet, 315
MoleculeDarkColorsGet, 272
MoleculeDarkColorsSet, 272
MoleculeDeleteHydrogens, 338
MoleculeExamine, 315
MoleculeGenerateCoords, 338
MoleculeGenerateCoordsFixed, 338
MoleculeGenerateCoordsFixedScoped, 339
MoleculeGet, 316
MoleculeHasComponents, 316
MoleculeMaxResidueGet, 316
MoleculeMergeScoped, 316
MoleculeNew, 339
MoleculeNewSubset, 316
MoleculeNewSubsetScoped, 316
MoleculeResidueNameSetScoped, 317
MoleculeResidueSet, 317
MoleculeRotate, 339
MoleculeSetProperty, 317
MoleculeShowfAntsyGet, 272
MoleculeShowfAntsySet, 272
MoleculeSizeCutoffGet, 317
MoleculeSizeCutoffSet, 317
MoleculeStyleGet, 272
MoleculeStyleGetScoped, 272
MoleculeStyleLargeGet, 272
MoleculeStyleLargeSet, 273
MoleculeStyleNucleicGet, 273
MoleculeStyleNucleicSet, 273
MoleculeStyleProteinGet, 273
MoleculeStyleProteinSet, 273
MoleculeStyleSet, 274
MoleculeStyleSetScoped, 274
MoleculeStyleToEnum, 274
MoleculeStyleToText, 274
MoleculeUpdate, 317
MonitorAngleCreate, 274
MonitorAngleDelete, 275
MonitorAngleExists, 275
MonitorColorGet, 275
MonitorColorSet, 275
MonitorDeleteScoped, 275
MonitorDistanceCreate, 275
MonitorDistanceDelete, 275
MonitorDistanceExists, 275
MonitorSphereCreate, 276
MonitorsVisible, 276
MonitorsVisibleDelete, 277

MonitorTorsionCreate, 276
MonitorTorsionDelete, 276
MonitorTorsionExists, 276

N

NameGet, 317
NameSet, 318

O

ObjectNameGet, 363
ObjectNameSet, 363
ObservableBool, 181
ObservableFloat, 181
ObservableInt, 181
ObservableKey, 182
ObservableOEKey, 363
ObservableString, 182
ObservableUInt, 182
ObservableUpdate, 182
ObservableVecFloat, 182
ObservableVecInt, 182
ObservableVecString, 182
ObservableVecUInt, 182
OEFuseKeyGroups, 318
OEGetKey, 360
OEKeyIterToVector, 318
OEKeyToID, 360
OEKeyToLabelString, 360
OEKeyToParentID, 360
OEKeyToSourceID, 361
OEPyClearActive, 361
OEPyClearLocked, 361
OEPyClearMarked, 361
OEPyClearSelected, 361
OEPyClearVisible, 361
OEPyGetActive, 361
OEPyGetIDForKey, 361
OEPyGetSelectedAtoms, 361
OEPyHide, 362
OEPyHideNone, 362
OEPyHideOthers, 362
OEPyIsActive, 362
OEPyIsLocked, 362
OEPyIsMarked, 362
OEPyIsSelected, 362
OEPyIsTrulyVisible, 362
OEPyIsVisible, 362
OEPySetActive, 363
OEPySetLocked, 363
OEPySetMarked, 363
OEPySetSelected, 363
OEPySetVisible, 363
Open, 183
OpenState, 183

P

PaneActivated, 277
PasteMolecules, 183
Pick, 216
PickAgain, 216
PopIgnoreHint, 217
PopState, 364
PopupAddButton, 217
PopupAddLabel, 364
PopupAddRadioButton, 217
PopupAddSeparator, 217
PopupAddSetupFunc, 217
PopupAddSubmenu, 217
PopupAddToggleButton, 217
PopupBeginRadioGroup, 218
PopupDisplayName, 218
PopupEnableItem, 218
PopupEnableItemCommand, 218
PopupEndRadioGroup, 218
PopupRemoveAll, 218
PopupRemoveItem, 218
PreferenceDump, 183
PreferenceGetBool, 183
PreferenceGetColor, 183
PreferenceGetDouble, 183
PreferenceGetFloat, 184
PreferenceGetInt, 184
PreferenceGetString, 184
PreferenceGetVBool, 184
PreferenceGetVDouble, 184
PreferenceGetVFloat, 184
PreferenceGetVInt, 184
PreferenceIsBool, 184
PreferenceIsColor, 185
PreferenceIsDouble, 185
PreferenceIsFloat, 185
PreferenceIsInt, 185
PreferenceIsSet, 185
PreferenceIsString, 185
PreferenceIsVBool, 185
PreferenceIsVDouble, 185
PreferenceIsVFloat, 185
PreferenceIsVInt, 186
PreferenceMark, 186
PreferenceRemove, 186
PreferenceRestore, 186
PreferencesEdit, 187
PreferenceSetBool, 186
PreferenceSetColor, 186
PreferenceSetCommand, 186
PreferenceSetDouble, 186
PreferenceSetFloat, 186
PreferenceSetInt, 187
PreferenceSetString, 187

PreferenceSetVBool, 187
PreferenceSetVDouble, 187
PreferenceSetVFloat, 187
PreferenceSetVInt, 187
PreferenceValidateCommands, 187
ProgressbarUpdate, 219
PromptAtomicNum, 219
PromptBookmarkAnimationTime, 254
PromptColor, 219
PromptError, 219
PromptFilename, 219
PromptFilenames, 219
PromptFloat, 219
PromptFont, 220
PromptFragment, 222
PromptID, 220
PromptIDs, 221
PromptIDWriteFilters, 220
PromptInteger, 221
PromptKey, 221
PromptKeys, 221
PromptMessage, 221
PromptModeGet, 221
PromptModeSet, 221
PromptMolecule, 222
PromptMoleculeSplit, 222
PromptMulti, 222
PromptQuery, 222
PromptResponseBool, 222
PromptResponseCanceled, 222
PromptResponseColor, 223
PromptResponseFloat, 223
PromptResponseInt, 223
PromptResponseKey, 223
PromptResponseKeys, 223
PromptResponseString, 223
PromptResponseUInt, 223
PromptResponseVectInt, 224
PromptResponseVectMulti, 224
PromptResponseVectString, 224
PromptResponseVectUInt, 224
PromptSaveFilename, 224
PromptSmiles, 225
PromptString, 225
PromptStringListFromString, 225
PromptStringListToString, 225
PromptYesNo, 225
PromptYesNoSetDefault, 225
PropertyTypeGet, 318
ProteinColorByBFactor, 277
ProteinColorByBFactorScoped, 277
PushIgnoreHint, 226

Q

Quit, 188

R

Redo, 188
RedoTo, 188
ResidueColorPaletteUpdate, 277
ResidueDarkColorsGet, 277
ResidueDarkColorsSet, 277
ResidueDefaultColorGet, 278
ResidueDefaultColorSet, 278
ResponseVectMulti, 364
RibbonClearColorScoped, 278
RibbonColorSetScoped, 278
RibbonCrossResolutionGet, 278
RibbonCrossResolutionSet, 278
RibbonGapGet, 278
RibbonGapSet, 279
RibbonHeightScaleGet, 279
RibbonHeightScaleSet, 279
RibbonRadiusGet, 279
RibbonRadiusSet, 279
RibbonResolutionGet, 279
RibbonResolutionSet, 279
RibbonSplineTypeGet, 279
RibbonSplineTypeSet, 279
RibbonStyleGet, 280
RibbonStyleSet, 280
RibbonStyleSetScoped, 280
RibbonWidthScaleGet, 280
RibbonWidthScaleSet, 280

S

Save, 188
SaveMiniState, 188
SaveState, 188
SaveStateFilter, 188
SceneDrawActiveBorderGet, 280
SceneDrawActiveBorderSet, 280
SceneMatrixModeGet, 280
SceneMatrixModeSet, 281
ScratchClear, 200
ScratchGet, 200
ScratchInsert, 200
ScratchList, 364
ScratchSet, 200
SDataGet, 364
SDataHas, 364
SDataSet, 364
SDDataGet, 318
SDDataHas, 318
SDDataSet, 318
Select, 200

SelectAllMolecules, 200
SelectByQuery, 200
SelectInvert, 201
SelectionColorBlendFactorGet, 281
SelectionColorBlendFactorSet, 281
SelectKeys, 201
SelectOERID, 201
SelectResidues, 201
SelectScoped, 201
SelectWithin, 201
SelectWithout, 201
SetProgressbar, 226
SettingsGetBool, 188
SettingsGetFloat, 189
SettingsGetInt, 189
SettingsGetString, 189
SettingsRestore, 189
SettingsSetBool, 189
SettingsSetFloat, 189
SettingsSetInt, 189
SettingsSetString, 189
SettingsSync, 189
ShowESGridScoped, 281
ShowSurface, 281
ShowSurfaceScoped, 281
SketcherInputSet, 339
SketcherLookupFunctionSet, 339
SpreadsheetAddColumn, 344
SpreadsheetColumnColorerSet, 346
SpreadsheetColumnController, 346
SpreadsheetColumnFontGet, 346
SpreadsheetColumnFontSet, 346
SpreadsheetColumnReadonly, 347
SpreadsheetColumnSigFigGet, 347
SpreadsheetColumnSigFigSet, 347
SpreadsheetCommitChanges, 347
SpreadsheetCopy, 347
SpreadsheetCreateColumnExpression, 347
SpreadsheetCreateFilter, 347
SpreadsheetCurrent, 365
SpreadsheetCurrentGet, 348
SpreadsheetCurrentSet, 348
SpreadsheetData, 348
SpreadsheetDeleteColumn, 348
SpreadsheetDisableUpdates, 365
SpreadsheetEditableGet, 348
SpreadsheetEditableSet, 348
SpreadsheetEnableUpdates, 365
SpreadsheetFilter, 348
SpreadsheetFromList, 349
SpreadsheetGetColumn, 349
SpreadsheetGetCurrentRow, 349
SpreadsheetGetIDForRow, 349
SpreadsheetGetImageStreamAtRow, 349
SpreadsheetGetKeyForRow, 349
SpreadsheetGetNumRows, 349
SpreadsheetGetRowForKey, 350
SpreadsheetGetSpreadsheets, 350
SpreadsheetHeaders, 350
SpreadsheetHideColumn, 350
SpreadsheetHideTab, 350
SpreadsheetImport, 350
SpreadsheetLingoSimSort, 350
SpreadsheetLoadFilter, 350
SpreadsheetMarkHighlighted, 365
SpreadsheetMolNumberFunction, 351
SpreadsheetMolStringFunction, 351
SpreadsheetMoveColumn, 352
SpreadsheetNumRows, 352
SpreadsheetPromptColumnExpression, 352
SpreadsheetPromptExport, 352
SpreadsheetPromptFilter, 352
SpreadsheetPromptFormat, 352
SpreadsheetPromptGraphemeOpts, 352
SpreadsheetPromptImport, 352
SpreadsheetPromptSort, 353
SpreadsheetRemoveTab, 353
SpreadsheetRowHeightGet, 353
SpreadsheetRowHeightSet, 353
SpreadsheetSetData, 353
SpreadsheetSetExpression, 353
SpreadsheetSetRowData, 354
SpreadsheetShowAllColumns, 354
SpreadsheetShowAllTabs, 354
SpreadsheetShowColumn, 354
SpreadsheetShowStats, 354
SpreadsheetShowStatsGet, 354
SpreadsheetShowStatsSet, 354
SpreadsheetShowTab, 355
SpreadsheetSort, 355
SpreadsheetUpdateContents, 365
StateMark, 190
StatePop, 190
Subset, 201
SurfaceAdd, 319
SurfaceAlterTransparency, 281
SurfaceBestFloodScoped, 319
SurfaceCheckIn, 319
SurfaceCheckOut, 319
SurfaceColorBy, 282
SurfaceColorByScoped, 282
SurfaceColorGet, 282
SurfaceColorGetScoped, 282
SurfaceColorSet, 283
SurfaceColorSetScoped, 283
SurfaceCreate, 319
SurfaceCreateScoped, 320
SurfaceCropDistance, 320

SurfaceCropDistanceFrom, 320
SurfaceCropScribedScoped, 320
SurfaceCropUnscribedScoped, 320
SurfaceDelete, 320
SurfaceGenerateBox, 320
SurfaceGenerateSphere, 321
SurfaceGenerateSpline, 321
SurfaceGrowTriangle, 283
SurfaceLineWidthGet, 283
SurfaceLineWidthSet, 283
SurfacePickTriangle, 321
SurfacePotentialColorAuto, 365
SurfacePotentialColorValuesSet, 365
SurfaceProbeRadiusGet, 321
SurfaceProbeRadiusSet, 321
SurfaceResolutionGet, 321
SurfaceResolutionSet, 321
SurfaceRestoreScoped, 322
SurfaceScribeScoped, 322
SurfaceSetPotentialFromGrid, 322
SurfaceSetPotentialFromGridScoped, 322
SurfaceStyleGet, 283
SurfaceStyleGetScoped, 283
SurfaceStyleSet, 284
SurfaceStyleSetScoped, 284
SurfaceTransparencySet, 284
SurfaceTransparencySetScoped, 284
SurfaceVertexFloodScoped, 284
SurfaceVolume, 322
SymmetryColorModeGet, 284
SymmetryColorModeSet, 284
SymmetryNumOperators, 322
SymmetryOperatorEnabledGet, 322
SymmetryOperatorEnabledSet, 322
SymmetryRadiusGet, 323
SymmetryRadiusSet, 323
SymmetryRealize, 323

T

TitleDefaultColorGet, 285
TitleDefaultColorSet, 285
TitlesDrawAboveGet, 285
TitlesDrawAboveSet, 285
ToolbarAddSeparator, 226
ToolbarAddToggle, 226
ToolbarBeginRadio, 226
ToolbarButtonEnableGet, 226
ToolbarButtonEnableSet, 227
ToolbarCreate, 227
ToolbarEnabledGet, 227
ToolbarEnabledSet, 227
ToolbarEndRadio, 227
ToolbarGetAll, 227
ToolbarItemCheckedGet, 227
ToolbarItemCheckedSet, 227
ToolbarItemUpdate, 228
ToolbarItemVisibleGet, 228
ToolbarItemVisibleSet, 228
ToolbarRemove, 228
ToolbarUpdate, 228
ToolbarVisibleGet, 228
ToolbarVisibleSet, 228
TransparencySet, 285
TransparencySetScoped, 285

U

Undo, 190
UndoHint, 190
UndoMark, 190
UndoTo, 190
UpdateRedo, 190
UpdateStyleWidget, 228
UpdateUndo, 229

V

VFCopyFile, 365
VFGetMol, 365
VFSleep, 190
ViewerActiveAnnotation, 229
ViewerActiveAnnotationBackgroundColorGet, 229
ViewerActiveAnnotationBackgroundColorSet, 229
ViewerActiveAnnotationClose, 229
ViewerActiveAnnotationFontGet, 229
ViewerActiveAnnotationFontSet, 229
ViewerActiveAnnotationForegroundColorGet, 229
ViewerActiveAnnotationForegroundColorSet, 229
ViewerActiveAnnotationVisible, 230
ViewerActiveDataFontSizeGet, 230
ViewerActiveDataFontSizeSet, 230
ViewerActiveDataHide, 230
ViewerActiveDataShow, 230
ViewerActiveDataVisible, 230
ViewerAmbientLightGet, 285
ViewerAmbientLightSet, 285
ViewerAmbientMaterialGet, 286
ViewerAmbientMaterialSet, 286
ViewerAnimate, 286
ViewerAnimateTo, 286
ViewerAntialiasGet, 287
ViewerAntialiasSet, 287
ViewerAutoCenterGet, 287
ViewerAutoCenterPanesGet, 287
ViewerAutoCenterPanesSet, 287
ViewerAutoCenterSet, 287
ViewerAutoFitGet, 287
ViewerAutoFitSet, 287
ViewerBackgroundColorGet, 288
ViewerBackgroundColorSet, 288

ViewerBookmarkLoad, 288
ViewerBookmarksGetAnimated, 288
ViewerBookmarksGetAnimationTime, 288
ViewerBookmarksSetAnimated, 288
ViewerBookmarksSetAnimationTime, 288
ViewerBookmarkWidget, 230
ViewerBookmarkWidgetFontSizeGet, 230
ViewerBookmarkWidgetFontSizeSet, 231
ViewerBookmarkWidgetHide, 231
ViewerBookmarkWidgetShow, 231
ViewerButtonImage, 231
ViewerCenterAndRadiusGet, 288
ViewerCenterAndRadiusSet, 288
ViewerCenterGet, 289
ViewerCenterSet, 289
ViewerCenterSetScoped, 289
ViewerClick, 231
ViewerCursorSet, 231
ViewerDepict, 232
ViewerDepictionAntiAliasGet, 232
ViewerDepictionAntiAliasSet, 232
ViewerDepictionHeightGet, 232
ViewerDepictionLineWidthGet, 232
ViewerDepictionLineWidthSet, 233
ViewerDepictionSizeSet, 233
ViewerDepictionWidthGet, 233
ViewerDepthcueEndGet, 289
ViewerDepthcueEndSet, 289
ViewerDepthCueFollowsSlab, 289
ViewerDepthcueGet, 290
ViewerDepthcueSet, 290
ViewerDepthcueStartGet, 290
ViewerDepthcueStartSet, 290
ViewerDiffuseLightGet, 290
ViewerDiffuseLightSet, 290
ViewerDiffuseMaterialGet, 290
ViewerDiffuseMaterialSet, 290
ViewerDrawDepictionsGet, 291
ViewerDrawDepictionsSet, 291
ViewerExportPOVRAY, 191
ViewerFit, 291
ViewerFontSizeGet, 291
ViewerFontSizeSet, 291
ViewerForwardGet, 291
ViewerGetShowObjectToolbar, 299
ViewerLabel, 233
ViewerLabelDialog, 233
ViewerLightPositionGet, 292
ViewerLightPositionSet, 292
ViewerLODGet, 291
ViewerLODSet, 291
ViewerLookAt, 292
ViewerMirrorSlabsGet, 292
ViewerMirrorSlabsSet, 292
ViewerMouseClick, 233
ViewerMouseDoubleClick, 234
ViewerMouseFunctionGet, 235
ViewerMouseFunctionNameGet, 235
ViewerMouseFunctionSet, 235
ViewerMouseMap, 235
ViewerMouseMove, 236
ViewerMouseOutsideAwareGet, 237
ViewerMouseOutsideAwareSet, 237
ViewerMouseReset, 237
ViewerMouseSensitivityGet, 238
ViewerMouseSensitivitySet, 238
ViewerMouseWheel, 238
ViewerMove, 239
ViewerNiceFontsGet, 292
ViewerNiceFontsSet, 292
ViewerOrientationGet, 292
ViewerOrientationSet, 293
ViewerPickSurfaceTrianglesGet, 239
ViewerPickSurfaceTrianglesSet, 239
ViewerProgress, 239
ViewerProjectorModeGet, 293
ViewerProjectorModeSet, 293
ViewerRadiusGet, 293
ViewerRadiusSet, 293
ViewerRecenter, 293
ViewerRemoveWidget, 239
ViewerReprobeStereo, 293
ViewerRotate, 293
ViewerScaleGet, 294
ViewerScaleSet, 294
ViewerScreenshot, 191
ViewerSetShowObjectToolbar, 299
ViewerShininessMaterialGet, 294
ViewerShininessMaterialSet, 294
ViewerShowActiveBorderGet, 294
ViewerShowActiveBorderSet, 294
ViewerShowGridGet, 294
ViewerShowGridSet, 294
ViewerShowTrackballGuideSet, 295
ViewerSlabEnableGet, 295
ViewerSlabEnableSet, 295
ViewerSlabFarGet, 295
ViewerSlabFarSet, 295
ViewerSlabNearGet, 295
ViewerSlabNearSet, 295
ViewerSlabWidthGet, 295
ViewerSlabWidthSet, 296
ViewerSpecularMaterialGet, 296
ViewerSpecularMaterialSet, 296
ViewerStereoAngleGet, 296
ViewerStereoAngleSet, 296
ViewerStereoCrossEyedGet, 296
ViewerStereoCrossEyedSet, 296

ViewerStereoEnableGet, 296
ViewerStereoEnableSet, 297
ViewerStereoSeparationGet, 297
ViewerStereoSeparationSet, 297
ViewerStereoStyleGet, 297
ViewerStereoStyleSet, 297
ViewerStyleControlVisibleGet, 297
ViewerStyleControlVisibleSet, 297
ViewerSupportsHWStereo, 297
ViewerTextBox, 239
ViewerTextFontGet, 298
ViewerTextFontSet, 298
ViewerTextScaleGet, 298
ViewerTextScaleSet, 298
ViewerToggleRenderFeatures, 298
ViewerTranslateX, 298
ViewerTranslateY, 298
ViewerTranslateZ, 298
ViewerUpGet, 299
ViewerUseDisplayListGet, 299
ViewerUseDisplayListSet, 299
ViewerUseInertiaGet, 239
ViewerUseInertiaSet, 240
ViewerUseKeyMapGet, 240
ViewerUseKeyMapSet, 240
ViewerUseSystemFontsGet, 299
ViewerUseSystemFontsSet, 299
ViewerWheel, 240
ViewerWidgetUpdate, 240
Visible, 202
VisibleScoped, 202
VisualizeCommandScopeGet, 202
VisualizeCommandScopeSet, 202

W

WaitBegin, 240
WaitEnd, 240
WindowEnabledGet, 240
WindowEnabledSet, 241
WindowMenuUpdate, 241
WindowRegisterHotkey, 241
WindowVisibleGet, 241
WindowVisibleSet, 241
WriteHistory, 191

X

XRayAutoMapCalculationPrefsSet, 323
XRayCalculateMap, 323
XRayCalculatePhases, 323
XRayGetCell, 323
XRayGetSpaceGroup, 324
XRayMTZColumnNamesCurrentDefaultsGet, 324
XRayMTZColumnNamesGet, 324
XRayMTZColumnNamesSet, 324
XRayMTZColumnNamesStandardDefaultsGet, 324
XRaySetCrystalParams, 324
XRayValidMaptypes, 325