

# **User Manual for wxCLIPS 1.63**

Julian Smart

January 1997

## Contents

<b>1. Introduction .....</b>	<b>1</b>
1.1. The relationship between CLIPS, wxCLIPS and wxWindows .....	1
1.2. Other products by Julian Smart.....	2
1.3. Acknowledgements and disclaimer .....	2
<b>2. Installing wxCLIPS .....</b>	<b>4</b>
<b>3. Running wxCLIPS .....</b>	<b>5</b>
<b>4. Compiling wxCLIPS .....</b>	<b>6</b>
4.1. Using wxCLIPS as a library.....	7
4.2. The wxCLIPS type system .....	8
4.3. Changes made to CLIPS .....	8
<b>5. Using CLIPS for building GUIs.....</b>	<b>10</b>
5.1. The wxCLIPS development window .....	10
5.2. Overview of wxCLIPS GUI functionality.....	10
5.3. Using wxCLIPS functions.....	11
5.4. Development and debugging .....	16
5.5. Packaging your application .....	16
<b>6. wxCOOL class reference .....</b>	<b>18</b>
6.1. wxApplication is-a wxObject.....	18
6.2. wxBitmap is-a wxObject.....	18
6.3. wxBrush is-a wxObject.....	19
6.4. wxButton is-a wxItem .....	20
6.5. wxCanvas is-a wxWindow.....	21
6.6. wxCheckBox is-a wxItem .....	23
6.7. wxChoice is-a wxItem .....	23
6.8. wxClient is-a wxObject.....	25
6.9. wxCommandEvent is-a wxEvent.....	25
6.10. wxConnection is-a wxObject .....	26
6.11. wxCursor is-a wxBitmap.....	29
6.12. wxDatabase is-a wxObject.....	31
6.13. wxDate is-a wxObject .....	33
6.14. wxDC is-a wxObject.....	39
6.15. wxDialogBox is-a wxPanel .....	44
6.16. wxEvent is-a wxObject.....	46
6.17. wxEvtHandler is-a wxObject.....	47

6.18. wxFont is-a wxObject.....	47
6.19. wxFrame is-a wxWindow.....	48
6.20. wxHelpInstance is-a wxObject.....	51
6.21. wxGauge is-a wxItem.....	52
6.22. wxGroupBox is-a wxItem .....	53
6.23. wxIcon is-a wxBitmap .....	53
6.24. wxKeyEvent is-a wxEvent .....	54
6.25. wxListBox is-a wxItem.....	55
6.26. wxMemoryDC is-a wxCanvasDC.....	58
6.27. wxMenu is-a wxWindow .....	58
6.28. wxMenuBar is-a wxWindow.....	60
6.29. wxMessage is-a wxItem .....	61
6.30. wxMetaFile is-a wxObject.....	61
6.31. wxMetaFileDC is-a wxDC.....	62
6.32. wxMouseEvent is-a wxEvent.....	63
6.33. wxMultiText is-a wxText .....	65
6.34. wxObject.....	65
6.35. wxPanel is-a wxCanvas .....	66
6.36. wxItem is-a wxWindow .....	68
6.37. wxPen is-a wxObject.....	69
6.38. wxPostScriptDC is-a wxDC .....	69
6.39. wxPrinterDC is-a wxDC.....	70
6.40. wxRadioBox is-a wxItem .....	71
6.41. wxRecordSet is-a wxObject .....	72
Specify the name of the table you want to use. 6.42. wxServer is-a wxObject .....	79
6.43. wxSlider is-a wxItem .....	80
6.44. wxText is-a wxItem .....	81
6.45. wxTextWindow is-a wxWindow.....	82
6.46. wxTimer is-a wxObject.....	84
6.47. wxToolBar is-a wxPanel.....	84
6.48. wxWindow is-a wxEvtHandler .....	88
<b>7. wxCLIPS function groups.....</b>	<b>92</b>
7.1. How to use this reference .....	92
7.2. Application.....	92
7.3. Bitmap .....	93
7.4. Brush.....	95
7.5. Button.....	95
7.6. Canvas .....	96
7.7. Checkbox .....	100

7.8. Choice .....	100
7.9. Client .....	102
7.10. Colour .....	103
7.11. Command event .....	103
7.12. Connection .....	104
7.13. Cursor .....	106
7.14. Database .....	107
7.15. Date .....	109
7.16. Device context .....	115
7.17. Dialog box .....	121
7.18. Event .....	122
7.19. Font .....	123
7.20. Frame .....	123
7.21. Help .....	126
7.22. HWND functions .....	127
7.23. Gauge .....	128
7.24. Grid .....	129
7.25. Groupbox .....	136
7.26. Html .....	137
Opens a URL (not yet functioning).7.27. Icon .....	138
7.28. Instance table .....	140
7.29. Key event .....	140
7.30. Listbox .....	141
7.31. Memory device context .....	143
7.32. Menu .....	144
7.33. Menu bar .....	145
7.34. Message .....	146
7.35. Metafile .....	147
7.36. Metafile device context .....	148
7.37. Mouse event .....	148
7.38. Multi-line text .....	150
7.39. Object .....	153
7.40. Panel .....	154
7.41. Panel item .....	155
7.42. Pen .....	156
7.43. PostScript device context .....	156
7.44. Printer device context .....	157
7.45. Radiobox .....	157
7.46. Recordset .....	158
7.47. Server .....	165

7.48. Slider .....	165
7.49. Text .....	166
7.50. Text window.....	167
7.51. Timer .....	171
7.52. Toolbar .....	172
7.53. Window.....	175
7.54. Miscellaneous .....	179
<b>8. wxCLIPS classes by category .....</b>	<b>190</b>
8.1. Managed windows .....	190
8.2. Subwindows .....	190
8.3. Panel items.....	190
8.4. Convenience dialogs.....	191
8.5. Device contexts .....	191
8.6. Graphics device interface.....	192
8.7. Events .....	192
8.8. Interprocess communication .....	193
8.9. Database classes.....	193
8.10. File functions .....	193
8.11. Time-related functions.....	193
8.12. Noisy functions .....	194
8.13. Operating system functions.....	194
8.14. wxCLIPS environment functions.....	194
8.15. Data functions.....	194
<b>9. Topic overviews .....</b>	<b>196</b>
9.1. Window styles.....	196
9.2. Interprocess communication overview.....	199
9.3. Device context overview .....	202
9.4. Dialog box overview.....	202
9.5. Toolbar overview .....	203
9.6. Database classes overview.....	205
9.7. Grid overview.....	210
9.8. wxCOOL overview .....	213
9.9. Resource overview .....	215
<b>10. DDE commands that wxCLIPS recognizes .....</b>	<b>217</b>
<b>11. Change log.....</b>	<b>218</b>
11.1. Version 1.63 .....	218
11.2. Version 1.62 .....	218

11.3. Version 1.61 .....	218
11.4. Version 1.60 .....	218
11.5. Version 1.59 .....	218
11.6. Version 1.58 .....	218
11.7. Version 1.57 .....	218
11.8. Version 1.56 .....	219
11.9. Version 1.55 .....	219
11.10. Version 1.55 .....	219
11.11. Version 1.54 .....	219
11.12. Version 1.53 .....	219
11.13. Version 1.52 .....	220
11.14. Version 1.51 .....	220
11.15. Version 1.50 .....	220
11.16. Version 1.49 .....	220
11.17. Version 1.48 .....	220
11.18. Version 1.47 .....	220
11.19. Version 1.46 .....	220
11.20. Version 1.45 .....	221
11.21. Version 1.44 .....	221
11.22. Version 1.43 .....	221
11.23. Version 1.42 .....	221
11.24. Version 1.41 .....	221
11.25. Version 1.40 .....	221
11.26. Version 1.38 .....	222
11.27. Version 1.36 .....	222
11.28. Version 1.35 .....	222
11.29. Version 1.34 .....	222
11.30. Version 1.33 .....	223
11.31. Version 1.32 .....	223
11.32. Version 1.30 .....	223
11.33. Versions 1.00 to 1.20 .....	223
<b>Glossary.....</b>	<b>225</b>
API .....	225
Bit list .....	225
Callback .....	225
Canvas.....	225
DDE .....	225
Device context.....	225
Dialog box .....	225

---

Frame.....	225
GUI .....	225
Menu bar .....	226
Metafile .....	226
Open Look.....	226
Panel.....	226
Resource.....	226
Status line .....	226
XView .....	226
<b>Index.....</b>	<b>228</b>

## 1. Introduction

wxCLIPS was developed to enable CLIPS programmers to write portable, graphical programs which run under X and MS Windows. It is essentially CLIPS modified to work with an event driven style of programming, and a set of GUI functions. Its name reflects the fact that it is a CLIPS interface to wxWindows, a C++ GUI library also written by Julian Smart.

wxCLIPS supports CLIPS 6.0, and CLIPS 6.0 with fuzzy extensions, depending on how it is compiled.

wxCLIPS is really two entities:

1. A library of CLIPS functions to access a subset of wxWindows functionality.
2. A simple stand-alone development environment for developing wxWindows applications using these extra CLIPS functions.

The library can be used by any C++ program, to give it a tailoring language and interactive access to GUI functionality. The stand-alone wxCLIPS is a simple development interface making use of the library.

wxCLIPS is fundamentally a set of functions, rather than COOL objects, since C-based user extensions are restricted to functions. However, there is now a set of classes called *wxCOOL* (page 213) which encapsulates much of the wxCLIPS functionality in a properly object-oriented manner.

The wxCLIPS extensions to CLIPS were written by Julian Smart of the Artificial Intelligence Applications Institute, University of Edinburgh. You can get the latest version of the Windows and Sun Open Look and Motif binaries and source from the AIAI ftp site:

**<http://www.aiai.ed.ac.uk/~jacs/wxclips.html> (WWW)**  
**<ftp.aiai.ed.ac.uk/pub/packages/wxclips> (FTP)**

### 1.1. The relationship between CLIPS, wxCLIPS and wxWindows

wxWindows is a C++ class library for multi-platform development, developed at the Artificial Intelligence Applications Institute by Julian Smart and available free of charge.

CLIPS is NASA's expert system shell, allowing rule-based, functional and object-oriented programming in the one, portable system. It is written in C and is effectively free of charge.

wxCLIPS is the union of wxWindows and CLIPS -- a set of CLIPS functions to access a large portion of wxWindows functionality. It is available in three forms: as a library, within other products, or as an executable with a simple front end.

To fully understand wxCLIPS, it will help to read the wxWindows documentation. This is available from AIAI's anonymous ftp site in PostScript, wxHelp, and Windows Help format. The ftp site is <ftp.aiai.ed.ac.uk>, and wxWindows is in `/pub/packages/wxwin`.

This document only deals with functions added to standard CLIPS. There are PostScript, Word for Windows and ASCII files containing NASA's CLIPS documentation, some of which will be required for serious wxCLIPS development.



For further information, please contact Julian Smart at the following address:

**Dr Julian Smart**  
**149 Warrender Park Road**  
**Edinburgh**  
**EH9 1DT**  
**julian.smart@ukonline.co.uk**  
**<http://web.ukonline.co.uk/Members/julian.smart>**  
**Tel: 0131 466 0193**

## 1.2. Other products by Julian Smart

The following are available via the AIAI World Wide Web page or FTP site.

**Hardy** A superset of wxCLIPS, Hardy is a meta-CASE tool, or hypertext-based diagramming tool. It's good for rapidly developing applications that are heavily diagramming-oriented, such as business process modelling, object-oriented analysis and design, Petri nets, and knowledge acquisition. Hardy is free for academic and personal use.

**Tex2RTF** Translates a subset of LaTeX into WinHelp RTF, linear RTF, HTML and wxHelp format. Primarily a system for creating multi-format, multi-platform documentation, Tex2RTF can be useful for one-off LaTeX to word processor conversions if the contents are not too complex. MS Word 6 is supported rather well with contents page, indexing and simple bibliographic references. Tex2RTF is free.

**wxWindows** A free cross-platform C++ GUI toolkit with a growing following on the Internet and elsewhere. Most ftp-able AIAI applications are written in wxWindows.

### 1.2.1. About AIAI

The Artificial Intelligence Applications Institute was founded in 1985 as an off-shoot of the University of Edinburgh Department of Intelligence, as a technology transfer organization. AIAI works with many commercial clients as well as on government-funded projects. Past and present work includes planning and scheduling, enterprise decision support, egress modelling, toxic substances expert system, CASE tool development, oil well prospecting decision support, and intelligent database access.

Julian Smart joined the Decision Support Group at AIAI in 1991 and has been developing Hardy and wxWindows since 1992, with occasional forays into other areas. Julian left AIAI in 1996 and is available for freelance GUI and other design and programming. He is keen to promote free, multiplatform, Internet-based projects and the uptake of AI techniques into mainstream computing.

## 1.3. Acknowledgements and disclaimer

We are very grateful to NASA for producing and making available the CLIPS language, which we have used as an embedded language within wxCLIPS.

The National Research Council of Canada have added fuzzy extensions CLIPS, and Bob Orchard of NRCC has made wxCLIPS compatible with these.

We would also like to thank the AIAI staff and students who have helped find bugs in wxCLIPS.

All trademarks acknowledged.

Copyright (c) 1996 Julian Smart and Artificial Intelligence Applications Institute, The University of  
Edinburgh

Permission to use, copy, modify, and distribute this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice, author statement and this permission notice appear in all copies of this software and related documentation.

THE SOFTWARE IS PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR OTHERWISE, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL JULIAN SMART OR THE ARTIFICIAL INTELLIGENCE APPLICATIONS INSTITUTE OR THE UNIVERSITY OF EDINBURGH BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT OR CONSEQUENTIAL DAMAGES OF ANY KIND, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER OR NOT ADVISED OF THE POSSIBILITY OF DAMAGE, AND ON ANY THEORY OF LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

## 2. Installing wxCLIPS

The Windows versions of wxCLIPS come with an installation program which should be self-explanatory. However, there are a few issues.

1. Currently, Windows versions require ODBC 2.5. If you have a different version installed, you may experience problems. To upgrade to ODBC 2.5, please download the ODBC Desktop Driver Pack 3.0 from:  
`ftp.aiai.ed.ac.uk/pub/packages/wxwin/tools/odbc25.zip`  
or, if you have Visual C++ 4.0, you can find it in the `redist` directory of your VC++ 4.0 CD-ROM.
2. For platforms other than Windows, if there is no binary available you will need to recompile wxCLIPS. Please see *Compiling wxCLIPS* (page 6) for details.

### 3. Running wxCLIPS

You can run wxCLIPS with or without command line arguments. The command line arguments are as follows:

- `-clips file`: Specifies a CLIPS file to batch on startup.
- `-load file`: Specifies a CLIPS file to load on startup. The file should contain constructs only, unlike `-clips` which can execute constructs. `-load` is much faster.
- `-start`: Specifies that wxCLIPS should hide the development window and call the function *app-on-init* on startup.
- `-dir directory`: Specifies a directory to change to on startup.
- `-h`: Returns help on command line arguments (under UNIX only).

In addition, if you supply a CLIPS file as the first argument to wxCLIPS without any switches, wxCLIPS will batch the file, change to that directory, and call *app-on-init*. This way you can have a wxCLIPS association for `.clp` and simply run the file.

See also *Packaging your application* (page 16).

## 4. Compiling wxCLIPS

wxCLIPS comes with modified CLIPS source files for CLIPS 6.04, and CLIPS 6.04 with NRCC's fuzzy extensions.

If the source is not supplied with your version of wxCLIPS, download it from:

```
ftp.aiail.ed.ac.uk:/pub/packages/wxclips
```

The source archive should contain both wxclips and wxextend directories, which should be placed below the utils directory of your wxWindows installation (see below). The wxExtend library is a helper library for interfacing wxWindows to interpreted languages via C.

You need to obtain the full distribution of each version of CLIPS, including any patches provided by NASA or NRCC. Fuzzy CLIPS is available from:

```
ai.iit.nrc.ca:pub/fzclips
```

You also need the latest wxWindows distribution, available from:

```
ftp.aiail.ed.ac.uk:/pub/packages/wxwin/X.XX
```

where X.XX is a version number such as 1.66.

Copy the modified files from one of the wxCLIPS directories CLIPS6.0 or FUZZY into the distribution, and make a library version of CLIPS. Use the makefiles provided in the respective wxCLIPS directories, specifying the target `clips.lib` (or `clips32v.lib` for a MS VC++ 4.0-compiled library).

Next, compile wxWindows for your platform. Supported platforms are UNIX with Motif 1.2 or XView 3.1, and MS Windows (using Microsoft or Borland C++). Minor adjustments may be needed for your compiler and environment.

Now compile wxCLIPS, after editing the makefile in the wxCLIPS src directory to set appropriate pointers to the CLIPS directory.

Under DOS, a typical command line to make wxCLIPS with CLIPS 6.0 is:

```
nmake -f makefile.dos CLIPS6=1 wxclips.exe
or
nmake -f makefile.dos full
```

Or for a fuzzy version:

```
nmake -f makefile.dos CLIPS6=1 FUZZY=1 wxclips.exe
or
nmake -f makefile.dos fullfuzz
```

For a non-debugging version (which gives a smaller executable), delete wxclips.exe and invoke make with `FINAL=1` appended to the make command line.

Under UNIX, use `make -f makefile.unx` with target `xview` or `motif`. Then use 'strip' to remove debugging information from the executable.

## 4.1. Using wxCLIPS as a library

The library is used by including the header file `wx_cmds.h` and linking with `libcmds.a` or `wxcmds.lib`. In addition to the set of new CLIPS functions described in the reference section, there are a few C++ functions to help embed CLIPS.

### **::ClipsErrorFunction**

**void ClipsErrorFunction(char \*s)**

This function must be defined when using the wxCLIPS library. It will be called whenever a wxCLIPS error occurs to print out a message. The object **ClipsError** may be used with stream output operators to output error messages; the operators call the **ClipsErrorFunction** as appropriate.

### **::RouteCommand**

**void RouteCommand(char \*command, int showResult)**

Internal CLIPS command modified to be 're-entrant'. Takes a string containing a CLIPS command and executes it. This function cannot return a result from CLIPS on its own, but used in conjunction with the wxCLIPS function **return-result** and some predefined global variables the same effect can be achieved. For example,

```
wxFrame *main_frame = NULL;
clipsReturnType = clipsUNKNOWN;
RouteCommand("(return-result (app-on-init))");
if (clipsReturnType == clipsLONG)
{
    main_frame = (wxFrame *)wxGetTypedObject(clipsReturnLong,
wxTYPE_FRAME);
}
```

See the entry for **return-result** for more details.

### **::wxExecuteClipsFile**

**void wxExecuteClipsFile(char \*filename)**

Execute a CLIPS batch file.

### **::wxCleanWindows**

**void wxCleanWindows(void)**

Deletes wxWindows frames and dialog boxes created with CLIPS functions, allowing an application to be restarted from the wxCLIPS environment or the user's environment.

### **::wxInitClips**

**void wxInitClips(void)**

Call this instead of the normal CLIPS initialization function, to initialize CLIPS and the wxCLIPS library.

**::wxRouteNoEcho****void wxRouteNoEcho(char \*command)**

Calls RouteCommand, but ensures that no echoing of returned results will take place during the call.

**::wxUserFunctions****void wxUserFunctions(void)**

This function contains the wxCLIPS function definitions. You must call this function from within the **UserFunctions** function that the CLIPS manual specifies you must define. This allows you to define additional CLIPS functions.

**4.2. The wxCLIPS type system**

To communicate between C++ and CLIPS, wxCLIPS uses long integers to represent objects. For convenience, they are simply the addresses of the C++ objects. However, when these handles are passed back to C++, wxCLIPS cannot simply coerce these integers back to objects, since the object may not exist or the type may be wrong. The program would simply crash without an error message, which is clearly unacceptable.

Instead, there is an explicit type system which, in conjunction with a hash table for the objects, allows C++ implementations of CLIPS functions to check that an objects exists and to check that the type is suitable for the intended operation. The type does not have to be an exact match, as with C++, so long as the passed type is at least a subtype of the intended type. For example, say someone passed a **wxFrame** to the **window-centre** function. If a **wxWindow** has a virtual member function **Centre**, and **wxFrame** is a subtype of **wxWindow**, then it's permissible to coerce the **wxFrame** to a **wxWindow** and call the **Centre** function.

The helper library wxExtend provides the mapping between C++ wxWindows functionality and C functions/callbacks which can be used by interpreted languages such as wxCLIPS. Please refer to the wxExtend documentation for further details.

**4.3. Changes made to CLIPS**

The following notes describe some of the modifications to CLIPS made for compilation with a variety of compilers, to be compatible with wxWindows names, and to allow CLIPS to be called in an event-driven style.

**4.3.1. Accessing CLIPS C functions from C++**

The file `clipscpp.h` was created to declare a selection of CLIPS functions as `extern "C"` so that C++ code can access them.

### **4.3.2. Code modifications**

Please see the file `notes.txt` in the CLIPS6.0 and FUZZY subdirectories of the wxCLIPS source archive. Basically, `setup.h` has to be modified and `commlne.c` replaced.



## 5. Using CLIPS for building GUIs

This chapter is a short introduction to the capabilities of wxCLIPS for general CLIPS program development and for building GUIs (Graphical User Interfaces).

### 5.1. The wxCLIPS development window

wxCLIPS has a basic development environment, consisting of a window with menus, a small input window, an output text window, and a toolbar (Windows only).

Type into the text input window and press return or the **Go** button to send the input to CLIPS. Results and error messages are written to the output window. Don't expect to use standard input in your CLIPS programs, since the text window is read-only, and when your program finally runs it will probably have its own specialised interface for entering and displaying data.

The menu bar contains menus for loading and saving CLIPS files, and substitutes for various CLIPS functions, such as clearing and starting the rule interpreter and listing CLIPS constructs.

Under Windows, there is a toolbar to accelerate a few commonly-used commands such as loading, batching, running the rule system and running a GUI application. Depending on platform, you may also be able to copy a command from the development window and paste into the command line; under Windows, select the text with the mouse, then press the Copy tool, click on the command line, and press shift-insert.

There is no *in situ* editing of CLIPS files, but see *DDE commands that wxCLIPS recognizes* (page 217), which shows how some integration with an editor is now possible.

### 5.2. Overview of wxCLIPS GUI functionality

Using wxCLIPS, you can create *frames*, each with an optional *menu bar* comprising a number of *menus*. Within a frame, you can create one or more *subwindows*. Subwindows can be *panels*, *canvases* and *text subwindows*.

Panels are used to contain *panel items*, such as buttons, text-entry items and list boxes. You can place panel items at specific places, or omit the position information in which case a left-to-right, top-to-bottom layout policy is used.

The dialog box is a special form of panel which has its own frame, so instead of creating a frame and a panel, you just create a dialog box and populate it with panel items.

Canvases are used for drawing arbitrary graphics. To issue drawing commands, the handle of the canvas **device context** must be used, because device contexts can represent other devices also, such as printers and metafiles. The concepts of *pen* and *brush* allow separate setting of outline and fill attributes, and *fonts* may be created and assigned to the device context before drawing text. User input on a canvas is handled by registering **OnEvent** and **OnChar** handler functions, and using event accessor functions to find out about the input event.

Text subwindows are used for displaying text files or writing text programmatically, and in the case of the X version of wxCLIPS, editing the text directly.

Because your program may have to respond to feedback from the user, such as resizing or closing of frames, it is often necessary to install *event handlers* using *window-add-callback* (page 175) to tell wxCLIPS what function to call when a user event happens. You may wish to handle

frame resize events by calculating positions and sizes for subwindows and setting their sizes. If there is only one subwindow, wxCLIPS handles subwindow resizing automatically if no event handler is supplied.

Instead of using explicit resizing, you can use the **window-fit** function to fit a frame or panel around its contents. For example, you can create a frame, a panel, some items on the panel, and then fit the panel to the items, and finally fit the frame to the panel.

You don't always have to create a dialog box manually, and handle all the button presses and other events; there is a small number of convenience functions, such as *get-text-from-user*, *message-box*, *get-choice* and *file-selector*. These all block the program flow at the point at which they were called, until the user dismisses the dialog in some way. Your program can examine the value returned and carry on.

### 5.2.1. Restrictions

The CLIPS function interface to the portable GUI library wxWindows is extensive but not yet complete.

To some extent the development of wxCLIPS is driven by user demand, and what we have time to do. Suggestions or requests for extensions are welcome.

The major difference with standard CLIPS is that the (read) function does not take input from standard input, and in fact there is no concept of standard input in wxCLIPS. Such functions must be replaced with GUI calls such as *get-text-from-user*.

### 5.3. Using wxCLIPS functions

wxCLIPS is really an interface to a C++ library, wxWindows. In wxWindows, each GUI entity is an object, with member functions associated with the object's class.

wxCLIPS has to use functions, so most functions take an integer identifier used as the handle of the object and returned from its creation function. Function names are comprised of the type of GUI element followed by the actual function name (corresponding to the C++ member function). Examples are *check-box-set-value* and *frame-set-menu-bar*. In C++, these are defined as `wxCheckBox::SetValue` and `wxFrame::SetMenuBar` respectively.

Creation functions are of the form 'GUI element'-create, and return the ID of the new object.

The following program shows off some of the wxCLIPS GUI capabilities, by create a frame with a panel and various panel items,

```
;;; hello.clp
;;; Shows how a frame may be created, with a menu bar and
;;; panel, using low-level windows functions.
;;; Load using -clips <file> on the command line or using the Batch
;;; or Load commands from the CLIPS development window; type
;;; (app-on-init) to start.

(defglobal ?*main-frame* = 0)
(defglobal ?*subframe* = 0)
(defglobal ?*panel* = 0)
(defglobal ?*canvas* = 0)
(defglobal ?*text-win* = 0)
```

```
(defglobal ?*hand-cursor* = 0)

(defglobal ?*small_font* = 0)
(defglobal ?*green_pen* = 0)
(defglobal ?*black_pen* = 0)
(defglobal ?*red_pen* = 0)
(defglobal ?*cyan_brush* = 0)

(defglobal ?*xpos* = -1.0)
(defglobal ?*ypos* = -1.0)

(defglobal ?*bitmap* = 0)
(defglobal ?*button-bitmap* = 0)
(defglobal ?*icon* = 0)

;;; Sizing callback
(deffunction on-size (?id ?w ?h)
  (if (and (neq ?id 0) (neq ?*panel* 0) (neq ?*text-win* 0)) then
    (bind ?client-width (window-get-client-width ?id))
    (bind ?client-height (window-get-client-height ?id))
    (window-set-size ?*panel* 0 0 ?client-width (* ?client-height 0.666))
    (window-set-size ?*text-win* 0 (* ?client-height 0.666) ?client-width
  (/ ?client-height 3))
  )
)

;;; Utility function for drawing a bitmap
(deffunction draw-bitmap (?dc ?bitmap ?x ?y)
  (bind ?mem-dc (memory-dc-create))
  (memory-dc-select-object ?mem-dc ?bitmap)
  ; Blit the memory device context onto the destination device context
  (dc-blit ?dc ?x ?y (bitmap-get-width ?bitmap) (bitmap-get-height
?bitmap)
    ?mem-dc 0.0 0.0)
  (dc-delete ?mem-dc)
)

(deffunction draw-graphics (?dc)
  (if (> ?*bitmap* 0) then
    (draw-bitmap ?dc ?*bitmap* 0.0 250.0))

  (dc-set-font ?dc ?*small_font*)
  (dc-set-pen ?dc ?*green_pen*)
  (dc-draw-line ?dc 0.0 0.0 200.0 200.0)
  (dc-draw-line ?dc 200.0 0.0 0.0 200.0)

  (dc-set-pen ?dc ?*red_pen*)
  (dc-set-brush ?dc ?*cyan_brush*)
  (dc-draw-rectangle ?dc 100.0 100.0 100.0 50.0)
  (dc-draw-rounded-rectangle ?dc 150.0 150.0 100.0 50.0)

  (dc-set-clipping-region ?dc 150.0 150.0 100.0 50.0)
  (dc-draw-text ?dc "This text should be clipped within the rectangle"
150.0 170.0)
  (dc-destroy-clipping-region ?dc)

  (dc-draw-ellipse ?dc 250.0 250.0 100.0 50.0)
```

```
(dc-draw-spline ?dc (mv-append 50.0 200.0 50.0 100.0 200.0 10.0))
(dc-draw-line ?dc 50.0 230.0 200.0 230.0)
(dc-draw-text ?dc "This is a test string" 50.0 230.0)
)

;;; Painting callback
(deffunction on-paint (?id)
  (if (neq ?id 0) then
    (bind ?dc (canvas-get-dc ?id))
    (draw-graphics ?dc)
  )
)

(deffunction on-event (?canvas ?event)
  (bind ?dc (canvas-get-dc ?canvas))
  (dc-set-pen ?dc ?*black_pen*)
  (bind ?x (mouse-event-position-x ?event))
  (bind ?y (mouse-event-position-y ?event))
  (bind ?dragging (mouse-event-dragging ?event))
  (if (and (> ?*xpos* -1) (> ?*ypos* -1) (> ?dragging 0)) then
    (dc-draw-line ?dc ?*xpos* ?*ypos* ?x ?y)
  )
  (bind ?*xpos* ?x)
  (bind ?*ypos* ?y)
)

(deffunction on-close (?frame)
  (format t "Closing frame.%n")
  (window-delete ?*subframe*)
  (bind ?*panel* 0)
  (bind ?*text-win* 0)
  1)

(deffunction on-menu-command (?frame ?id)
  (switch ?id
    (case 200 then (message-box "CLIPS for wxWindows Demo
by Julian Smart (c) 1993" wxOK 1 0 "About wxWindows CLIPS Demo"))
    (case 3 then (if (on-close ?frame) then (window-delete ?frame)))
    (case 1 then
      (bind ?file (file-selector "Choose a text file to load"))
      (if (neq ?file "") then
        (text-window-load-file ?*text-win* ?file)))
    (case 4 then
      (bind ?dc (postscript-dc-create "" 1))
      (if (and (> ?dc 0) (= (dc-ok ?dc) 1)) then
        (if (= (dc-start-doc ?dc "Printing") 1) then
          (dc-start-page ?dc)
          (draw-graphics ?dc)
          (dc-end-page ?dc)
          (dc-end-doc ?dc)
        )
      )
      (if (> ?dc 0) then (dc-delete ?dc))
    )
  )
)
```

```
;;; Button callback
(deffunction frame-button-proc (?id)
  (bind ?parent (window-get-parent ?id))
  (bind ?grandparent (window-get-parent ?parent))
  (format t "Pressed button %d%n" ?id)
  (message-box "Hello")
)

;;; Text callback
(deffunction text-callback (?id)
  (bind ?event-id (panel-item-get-command-event))
  (if (eq "wxEVENT_TYPE_TEXT_ENTER_COMMAND" (event-get-event-type
?event-id)) then
    (format t "The text was %s%n" (text-get-value ?id))
  )
)

;;; Radiobox callback
(deffunction radio-box-callback (?id)
)

;;; Test program to create a frame
(deffunction app-on-init ()
  (unwatch all)
  (if (= ?*small_font* 0) then
    (bind ?*small_font* (font-create 10 wxSWISS wxNORMAL wxNORMAL 0))
    (bind ?*green_pen* (pen-create GREEN 1 wxSOLID))
    (bind ?*black_pen* (pen-create BLACK 1 wxSOLID))
    (bind ?*red_pen* (pen-create RED 3 wxSOLID))
    (bind ?*cyan_brush* (brush-create CYAN wxSOLID))
    (bind ?*hand-cursor* (cursor-create "wxCURSOR_HAND"))
    (if (eq "Windows 3.1" (get-platform)) then
      (bind ?*bitmap* (bitmap-load-from-file "wxwin.bmp"))
      (bind ?*button-bitmap* (bitmap-load-from-file "aiai.bmp"))
      (bind ?*icon* (icon-load-from-file "aiai.ico"
"wxBITMAP_TYPE_ICO"))
    )
  )

  (bind ?*main-frame* (frame-create 0 "Hello wxCLIPS!" -1 -1 500 460))
  (frame-create-status-line ?*main-frame*)
  (frame-set-status-text ?*main-frame* "Welcome to wxCLIPS")
  (if (> ?*icon* 0) then
    (frame-set-icon ?*main-frame* ?*icon*)
  )

  (window-add-callback ?*main-frame* OnSize on-size)
  (window-add-callback ?*main-frame* OnClose on-close)
  (window-add-callback ?*main-frame* OnMenuCommand on-menu-command)

  ;;; Make a menu bar
  (bind ?file-menu (menu-create))
  (menu-append ?file-menu 1 "&Load file")
  (menu-append ?file-menu 4 "&Print to PostScript")

  (bind ?pull-right (menu-create))
  (menu-append ?pull-right 100 "&Twips")
```

```
(menu-append ?pull-right 101 "&10th mm")

(menu-append ?file-menu 2 "&Scale picture" ?pull-right)
(menu-append-separator ?file-menu)
(menu-append ?file-menu 3 "&Quit")

(bind ?help-menu (menu-create))
(menu-append ?help-menu 200 "&About")

(bind ?menu-bar (menu-bar-create))
(menu-bar-append ?menu-bar ?file-menu "&File")
(menu-bar-append ?menu-bar ?help-menu "&Help")

(frame-set-menu-bar ?*main-frame* ?menu-bar)

;;; Make a panel and panel items
(bind ?*panel* (panel-create ?*main-frame* 0 0 530 250))
(panel-set-label-position ?*panel* wxVERTICAL)

(bind ?*text-win* (text-window-create ?*main-frame* 0 250 500 250))

(bind ?button (button-create ?*panel* frame-button-proc "A button"))
(if (> ?*button-bitmap* 0) then
  (bind ?bitmap-button (button-create-from-bitmap ?*panel* frame-
button-proc ?*button-bitmap*))
)
(bind ?text (text-create ?*panel* "text-callback" "A text item"
"Initial value" -1 -1 200 -1 "wxPROCESS_ENTER"))
(bind ?check (check-box-create ?*panel* "" "A check box"))

(panel-new-line ?*panel*)

(bind ?choice (choice-create ?*panel* "" "A choice item" -1 -1 -1 -1
(mv-append
  "One" "Two" "Three" "Four"))))
(choice-set-selection ?choice 0)

(message-create ?*panel* "Hello! A simple message")

(bind ?list (list-box-create ?*panel* "" "A list" 0 -1 -1 100 80))
(list-box-append ?list "Apple")
(list-box-append ?list "Pear")
(list-box-append ?list "Orange")
(list-box-append ?list "Banana")
(list-box-append ?list "Fruit")

(panel-new-line ?*panel*)

(bind ?slider (slider-create ?*panel* "" "A slider" 40 22 101 200))

(bind ?multi (multi-text-create ?*panel* "" "Multiline text" "Some
text" -1 -1 200 100))

(bind ?radiobox (radio-box-create ?*panel* "radio-box-callback"
"Radiobox" -1 -1 200 100
  (mv-append "1" "2" "3" "4" "5" "6") 2 "wxVERTICAL"))
(printout t "Radiobox id = " ?radiobox crlf)
```

```
; (window-fit ?*panel*)
; (window-fit ?*main-frame*)

(text-window-load-file ?*text-win* "hello.clp")
(bind ?*subframe* (frame-create 0 "Canvas Frame" 300 300 400 400))
(bind ?*canvas* (canvas-create ?*subframe* 0 0 400 400))
(window-set-cursor ?*canvas* ?*hand-cursor*)
(window-add-callback ?*canvas* OnPaint on-paint)
(window-add-callback ?*canvas* OnEvent on-event)
(canvas-set-scrollbars ?*canvas* 20 20 50 50 4 4)

(window-centre ?*main-frame* wxBOTH)

(window-show ?*main-frame* 1)
(window-show ?*subframe* 1)

?*main-frame*)
```

## 5.4. Development and debugging

Normally, you will write CLIPS files, and load (or batch) them into wxCLIPS from the **File** menu. If you find a bug, you can change the file in your editor and reload.

Note that there is a difference between batching and loading. The CLIPS *load* command only handles definitions, *not* function calls. However it does more error checking and is faster than batching. The purpose of batching is to allow function calls to be included in your files, and it is also the only way to automatically load CLIPS files from the command line (see below).

The best strategy is to have a small batch file which contains *load* commands to load the bulk of the application, plus a few function calls, and it is this file which is batched.

If you want to rerun a GUI application, but there are still windows lying around, you can use the **Clean up windows** menu option from the **Application** menu, or call the *clean-windows* function.

If you define a function called *app-on-init* (see next section) you can use the *Start application* menu item from the **Application** menu, which calls this function automatically.

Debugging can be made easier by typing `(watch all)` to get a trace of function calls.

## 5.5. Packaging your application

It is possible to make wxCLIPS load your application and start it up without the development window appearing. You need to define a function called *app-on-init*, which takes no arguments but must return the identifier of the top-level frame.

Now start your application as follows:

```
wxclips -clips load.clp -start
```

The file after the `-clips` switch is 'batched', which can be slow for a large file. Make the file small, with one or more calls to load definition files, followed by any function calls necessary (for example to register event handlers). If the file contains no CLIPS commands which should be executed immediately, use the `-load` switch instead.

The `-start` switch tells wxCLIPS to call the function *app-on-init*, and use the returned frame identifier as the top level window. The development window will not be displayed.

If you want wxCLIPS to start in a particular directory, you can use the `-dir` switch to `chdir` to a directory. This is useful to avoid the need for specifying absolute pathnames in your CLIPS code.

See also *Running wxCLIPS* (page 5).



## 6. wxCOOL class reference

See also *wxCOOL overview* (page 213)

This is the reference for the wxCOOL classes. With these functions, it is possible to create special-purpose user interfaces independent of platform. Currently these capabilities are supported under MS Windows, Open Look and Motif, except where stated.

### 6.1. wxApplication is-a wxObject

Not yet implemented.
----------------------

One object of this class can be created, and its implementation depends upon the C++ application hosting the wxCLIPS environment.

#### wxApplication on-char-hook

**bool** (on-char-hook *wxKeyEvent event*)

Under Windows only, all key strokes going to a dialog box or frame can be intercepted before being passed on for normal processing. This function takes the window id and event id, and should return 1 to override further processing, or 0 to do default processing. If the function returns 0, the on-char-hook message will be sent to the active window. See also *Key event* (page 140).

### 6.2. wxBitmap is-a wxObject

A bitmap is a rectangular array of pixels, possibly in colour. A bitmap can be created in memory, or loaded from an XBM file under X, or BMP file under Windows.

A bitmap can be drawn on a canvas by selecting it into a *wxMemoryDC* (page 58) object and using *dc-blit* (page 115). Bitmaps can also be used to create buttons; see *button-create-from-bitmap* (page 96).

#### wxBitmap bitmap-type

**string** bitmap-type

Indicates the type of bitmap file the bitmap is being loaded from.

May be one of:

- `wxBITMAP_TYPE_BMP`: Windows BMP (the default under Windows).
- `wxBITMAP_TYPE_XBM`: X monochrome bitmap (the default under X).
- `wxBITMAP_TYPE_GIF`: GIF bitmap (only under X).
- `wxBITMAP_TYPE_XPM`: XPM colour bitmap (under Windows and X if wxCLIPS has been compiled to include this option).
- `wxBITMAP_TYPE_RESOURCE`: Windows resource bitmap; unlikely to be used since the resources compiled into wxCLIPS cannot be changed from CLIPS.

**wxBitmap depth****long depth**

The depth of the bitmap (number of bits per pixel). Optionally initialize this if creating an in-memory bitmap; omitting it makes the depth default to the current display depth of the screen.

**wxBitmap filename****string filename**

If this slot is initialized on creation, the wxBitmap will be created from the given file. The slot *bitmap-type* must also be initialized, to indicate the type of bitmap file.

Defaults to the empty string.

**wxBitmap height****long height**

The height of the bitmap. Initialize this if creating an in-memory bitmap.

**wxBitmap width****long width**

The width of the bitmap. Initialize this if creating an in-memory bitmap.

**wxBitmap create****void (create)**

Creates a bitmap in memory, either blank or from an existing bitmap file. The programmer can draw into the bitmap by selecting it into a memory device context, for later drawing on an output device context such as a canvas device context.

The method of bitmap construction depends on the slots that are initialized when the instance is created. Here are some examples:

```
; Load from a BMP file
(make-instance [my-bitmap] of wxBitmap
  (filename "ai.ai.bmp") (bitmap-type "wxBITMAP_TYPE_BMP"))

; Create a 'blank' bitmap
(make-instance [my-bitmap] of wxBitmap
  (width 100) (height 100))
```

**6.3. wxBrush is-a wxObject**

A brush is an object that can be set for a *device context* (page 115) and determines the fill colour and style for subsequent drawing operations.

See also *wxPen* (page 69).

### **wxBrush colour**

#### **string colour**

The colour of the brush. It may be a *wxWindows* colour string such as "BLACK", "WHITE", "CYAN" etc.

### **wxBrush style**

#### **symbol style**

The style of the brush. It may be one of:

- *wxSOLID* (the default)
- *wxTRANSPARENT*
- *wxBDIAGONAL\_HATCH*
- *wxCROSSDIAG\_HATCH*
- *wxFDIAGONAL\_HATCH*
- *wxCROSS\_HATCH*
- *wxHORIZONTAL\_HATCH*
- *wxVERTICAL\_HATCH*

### **wxBrush create**

#### **void (create)**

Creates a brush for use in a device context. A brush must be set to fill graphic shapes.

The following slots must be initialized:

- *colour* is a *wxWindows* colour string such as "BLACK", "CYAN").
- *style* may be a value such as *wxSOLID* or *wxTRANSPARENT* (see *style* (page 20) for complete list).

## **6.4. wxButton is-a wxItem**

A *wxButton* is a rectangular control which can be placed on a *wxPanel* (page 66) to invoke a command.

### **wxButton bitmap**

#### **wxBitmap bitmap**

The bitmap associated with a *wxButton*, if being used as a bitmap button.

### **wxButton create**

**void (create)**

Creates a label or bitmap button on the given panel. If the *bitmap* slot is initialized, the button will be created from the bitmap. Otherwise, a text button will be created, using *label* for the label.

The following slots may be used in initializing a *wxButton* instance:

- *parent*: should be a *wxPanel* or *wxDialogBox*.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this button.
- *label*: for a text label button, must be a string.
- *bitmap*: for a bitmap label button, must be a *wxBitmap*.

When the button is pressed, the on-command message will be sent to the *wxButton*; if there is no default handler, it will be passed to the *wxButton*'s parent, and then to the parent's parent. See *wxCommandEvent* (page 25) for a list of event types associated with *wxCommandEvent*.

**6.5. wxCanvas is-a wxWindow**

A subwindow used for drawing arbitrary graphics. It must be the child of a *wxFrame* (page 48).

**wxCanvas dc****wxCanvasDC dc**

The device context handle belonging to the canvas. The device context must be retrieved before anything can be drawn on the canvas. If your drawing function is parameterized by a device context, you will be able to pass other types of device context to your drawing routine, such as PostScript and Windows metafile device contexts.

**wxCanvas create****bool (canvas-create)**

Creates a canvas for drawing graphics on. The following slots may be initialized.

- *parent*: should be a *wxFrame*.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this canvas.
- *style*: may be absent or a string style: see below.

The value of **style** can be a *bit list* of the following values:

**wxBORDER**     Gives the canvas a thin border (Windows 3 and Motif only).  
**wxRETAINED**   Gives the canvas a *wxWindows*-implemented backing store, making repainting

much faster but at a potentially costly memory premium (XView and Motif only).  
**wxBACKINGSTORE** Gives the canvas an X-implemented backing store (XView and Motif only). The X server may choose to ignore this request, whereas **wxRETAINED** is always implemented under X.

### **wxCanvas set-scrollbars**

**bool (set-scrollbars long x-unit-size long y-unit-size  
long x-length long y-length long x-page-length long y-page-length)**

Set the scrollbars for the given canvas. The first argument pair specifies the number of pixels per logical scroll unit, that is, the number of pixels to scroll when a scroll arrow is clicked. If either is zero or less, that scrollbar will not appear. The second pair specifies the size of the virtual canvas in logical scroll units. The third pair of arguments specify the number of scroll units per page, that is, the amount to scroll by when the scrollbar is page-scrolled (usually by clicking either side of the scrollbar handle).

### **wxCanvas scroll**

**bool (scroll long x-position long y-position)**

Scroll the canvas programmatically to the given scroll position. To convert from pixel position to scroll position, divide the pixel position by the scroll unit size you passed to *set-scrollbars* (page 22).

### **wxCanvas on-char**

**void (on-char wxKeyEvent event)**

Allows interception of key events. See also *wxKeyEvent* (page 54).

### **wxCanvas on-event**

**void (on-event wxMouseEvent event)**

Allows interception of mouse events. See also *wxMouseEvent* (page 63).

### **wxCanvas on-paint**

**void (on-paint)**

Override this handler to respond to paint events (sent when the canvas needs repainting).

### **wxCanvas on-size**

**void (on-size long width long height)**

The function is called with the canvas width and height when the canvas is resized.

## 6.6. wxCheckBox is-a wxItem

A wxCheckBox is a small box with a label, and can be in one of two states. It must be the child of a wxPanel (page 66).

### wxCheckBox value

#### bool value

The value of the checkbox (TRUE or FALSE).

### wxCheckBox create

#### void (create)

Creates a checkbox on the given panel. The following slots may be initialized.

- *parent*: should be a wxPanel or wxDialogBox.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this item.
- *label*: may be absent or a string name to label this item.
- *style*: reserved for future use.
- *value*: TRUE or FALSE.

## 6.7. wxChoice is-a wxItem

A wxChoice item is similar to a single-selection wxListBox (page 55) but normally only the current selection is displayed. It must be the child of a wxPanel (page 66).

### wxChoice values

#### multifield values

List of string values for initializing the wxChoice item.

### wxChoice create

#### bool (wxChoice create)

Creates a choice item on the given panel. A choice consists of a list of strings, one of which may be selected and displayed at any one time. The following slots may be initialized.

- *parent*: should be a wxPanel or wxDialogBox.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.

- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this item.
- *label*: may be absent or a string name to label this item.
- *style*: reserved for future use.
- *values*: a multifield list of strings.

Note that under Motif, it is recommended that the values are passed in this function, rather than using *append*, because of the nature of Motif. Otherwise, things are likely to be messed up.

### **wxChoice append**

**bool** (**append** string *item*)

Appends the string *item* to the choice.

### **wxChoice find-string**

**long** (**find-string** string *item*)

Searches for the given string and if found, returns the position ID of the string.

### **wxChoice clear**

**bool** (**clear**)

Clears all the strings from the choice item.

### **wxChoice get-selection**

**long** (**get-selection**)

Get the ID of the string currently selected.

### **wxChoice get-string-selection**

**string** (**get-string-selection**)

Get the string currently selected.

### **wxChoice set-selection**

**bool** (**set-selection** long *item-id*)

Sets the choice selection to the given item ID (numbered from zero).

### **wxChoice set-string-selection**

**bool** (**set-string-selection** **string** *item*)

Sets the selection by passing the appropriate item string.

### **wxChoice** **get-string**

**string** (**get-string** **long** *item-id*)

Gets the string associated with the given item ID.

## **6.8. wxClient is-a wxObject**

See also *Interprocess communication overview* (page 199)

Not yet tested.
-----------------

A client object represents the client side of a DDE conversation.

### **wxClient** **create**

**void** (**create**)

Creates a client object. You should override the on-make-connection handler to return an object of class derived from *wxConnection*, since various members of *wxConnection* must be overridden to intercept messages.

A connection is not made until *make-connection* (page 25) is called.

### **wxClient** **make-connection**

**wxConnection** (**make-connection** **string** *host* **string** *service* **string** *topic*)

Makes a connection to a server, returning an object of a user-defined derivative of *wxConnection* (page 26) if successful.

*host* is ignored under Windows, and should contain a valid internet host name under X.

*service* is a DDE service identifier (under X should contain a socket identifier).

*topic* is a topic name for this connection.

### **wxClient** **on-make-connection**

**wxConnection** (**on-make-connection**)

Should be overridden to return an object of the appropriate *wxConnection* class, whenever a connection is made. The base *wxConnection* class cannot be used because various members of *wxConnection* must be overridden in order to respond to messages from the server.

## **6.9. wxCommandEvent is-a wxEvent**



A `wxCommandEvent` is passed to a message handler when a panel item command is issued (usually by a user action).

The command event types are as follows:

- `wxEVENT_TYPE_BUTTON_COMMAND`
- `wxEVENT_TYPE_CHECKBOX_COMMAND`
- `wxEVENT_TYPE_CHOICE_COMMAND`
- `wxEVENT_TYPE_LISTBOX_COMMAND`
- `wxEVENT_TYPE_TEXT_COMMAND`
- `wxEVENT_TYPE_TEXT_ENTER_COMMAND`
- `wxEVENT_TYPE_MULTITEXT_COMMAND`
- `wxEVENT_TYPE_MENU_COMMAND`
- `wxEVENT_TYPE_SLIDER_COMMAND`
- `wxEVENT_TYPE_RADIOBOX_COMMAND`
- `wxEVENT_TYPE_SET_FOCUS`
- `wxEVENT_TYPE_KILL_FOCUS`

### **`wxCommandEvent` get-selection**

**long (get-selection)**

Returns the identifier selection corresponding to the selected item, for example a listbox or menu item.

### **`wxCommandEvent` is-selection**

**bool (is-selection)**

Returns 1 if the event was a selection event, 0 otherwise.

## **6.10. `wxConnection` is-a `wxObject`**

Not yet tested.
-----------------

See also *Connection overview* (page 200)

A `wxConnection` object has no creation function, since it is implicitly created when a connection is requested (one object at each side of the connection).

A connection object is used for initiating DDE commands and requests using functions such as `execute`, and it also has message handlers associated with it to respond to commands from the other side of the connection.

### **`wxConnection` service-name**

**string service-name**

Service name variable.

**wxConnection advise****bool (advise string *item* string *data*)**

Called by a server application to pass data to a client (for example, when a spreadsheet cell has been updated, and the client is interested in this value).

*item* is the name of the item, and *data* is a string representing the item's data.

Returns TRUE if successful, FALSE otherwise.

**wxConnection execute****bool (wxConnection execute string *data*)**

Called by a client application to execute a command in the server. Note there is no item in this command.

*data* is a string representing the item's data.

Returns TRUE if successful, FALSE otherwise.

To get a result from a server, you need to call *request* explicitly, since *execute* doesn't return data.

**wxConnection disconnect****bool (disconnect)**

Called by a client or server application to terminate this connection. After this call, the connection object is no longer valid.

Returns TRUE if successful, FALSE otherwise.

**wxConnection poke****bool (poke string *item* string *data*)**

Called by a client application to poke data into the server.

*item* is the name of the item, and *data* is a string representing the item's data.

Returns TRUE if successful, FALSE otherwise.

**wxConnection request****string (request string *item*)**

Called by a client application to request data from a server.

*item* is the name of the requested data item.

Returns a string representing the data if successful, the empty string otherwise.

### **wxConnection start-advise**

**bool (start-advise string *item*)**

Called by a client application to indicate interest in a particular piece of data in a server. The client connection should then receive OnAdvise messages when the data is updated in the server.

*item* is the name of the data item of interest.

Returns TRUE if the advise loop is allowed, FALSE otherwise.

### **wxConnection stop-advise**

**bool (stop-advise string *item*)**

Called by a client application to indicate a termination of interest in a particular piece of data in a server.

*item* is the name of the data item of interest.

Returns TRUE if successful, FALSE otherwise.

### **wxConnection on-advise**

**bool (on-advise string *topic* string *item* string *data*)**

Called on the client side of the connection, when the server side sends an *advise* message. Used for advising the client of a change in server data. Override this to intercept such messages.

### **wxConnection on-execute**

**bool (on-execute string *topic* string *data*)**

Called on the server side of the connection, when the client side sends an *execute* message. Used for implementing commands on the server side. Override this to implement command execution; you might wish to store the last result(s) to be returned when the client sends a *request* message.

### **wxConnection on-poke**

**bool (on-poke string *topic* string *item* string *data*)**

Called on the server side of the connection, when the client side sends a *poke* message. Used for poking data into a server. Override this to intercept such messages.

**wxConnection on-request****string (on-request string *topic* string *item*)**

Called on the server side of the connection, when the client side sends a *request* message. Used for getting information from a server. Override this to intercept such messages and return data back to the client.

**wxConnection on-start-advise****bool (on-start-advise string *topic* string *item*)**

Called on the server side of the connection, when the client side wishes to start an *advise* loop for the given topic and item. The server should respond with TRUE to accept this advise loop, FALSE otherwise.

**wxConnection on-stop-advise****bool (on-stop-advise string *topic* string *item*)**

Called on the server side of the connection, when the client side wishes to stop an *advise* loop for the given topic and item. The server should respond with TRUE to terminate this advise loop, FALSE otherwise.

**6.11. wxCursor is-a wxBitmap**

A cursor is a small bitmap used for representing the mouse pointer. It can be set for a particular subwindow, using *wxWindow set-cursor* (page 90), as a cue for what operations are possible in this window at this point in time.

**wxCursor cursor-name****string cursor-name**

A stock cursor name, one of the following:

- wxCURSOR\_ARROW
- wxCURSOR\_BULLSEYE
- wxCURSOR\_CHAR
- wxCURSOR\_CROSS
- wxCURSOR\_HAND
- wxCURSOR\_IBEAM
- wxCURSOR\_LEFT\_BUTTON
- wxCURSOR\_MAGNIFIER
- wxCURSOR\_MIDDLE\_BUTTON
- wxCURSOR\_NO\_ENTRY
- wxCURSOR\_PAINT\_BRUSH
- wxCURSOR\_PENCIL
- wxCURSOR\_POINT\_LEFT
- wxCURSOR\_POINT\_RIGHT
- wxCURSOR\_QUESTION\_ARROW

- `wxCURSOR_RIGHT_BUTTON`
- `wxCURSOR_SIZENESW`
- `wxCURSOR_SIZENS`
- `wxCURSOR_SIZENWSE`
- `wxCURSOR_SIZEWE`
- `wxCURSOR_SIZING`
- `wxCURSOR_SPRAYCAN`
- `wxCURSOR_WAIT`
- `wxCURSOR_WATCH`
- `wxCURSOR_BLANK`
- `wxCURSOR_CROSS_REVERSE` (X only)
- `wxCURSOR_DOUBLE_ARROW` (X only)
- `wxCURSOR_BASED_ARROW_UP` (X only)
- `wxCURSOR_BASED_ARROW_DOWN` (X only)

### **wxCursor x**

#### **long x**

The cursor hotspot x position (used only when loading a cursor from a file).

### **wxCursor y**

#### **long y**

The cursor hotspot y position (used only when loading a cursor from a file).

### **wxCursor create**

#### **void (create)**

Creates either a stock cursor (if *cursor-name* is non-nil) or a cursor loaded from a disk file (if *filename* and *bitmap-type* are non-nil).

Under X, the permitted cursor types in *bitmap-type* are:

- **`wxBITMAP_TYPE_XBM`** Load an X bitmap file

Under Windows, the permitted types are:

- **`wxBITMAP_TYPE_CUR`** Load a cursor from a .cur cursor file (only if `USE_RESOURCE_LOADING_IN_MSW` is enabled in `wx_setup.h`).
- **`wxBITMAP_TYPE_CUR_RESOURCE`** Load a Windows resource (as specified in the .rc file).
- **`wxBITMAP_TYPE_ICO`** Load a cursor from a .ico icon file (only if `USE_RESOURCE_LOADING_IN_MSW` is enabled in `wx_setup.h`). Specify x and y slot values.

Examples:

```
; Create a stock cursor
```

```
(bind ?cursor (make-instance (gensym*)
  of wxCursor (cursor-name "wxCURSOR_PENCIL")))

; Create a cursor from a .cur file
(bind ?cursor (make-instance (gensym*)
  of wxCursor (filename "figure.cur") (bitmap-type
"wxBITMAP_TYPE_CUR"))))

; Create a cursor from a .ico file
(bind ?cursor (make-instance (gensym*)
  of wxCursor (filename "figure.icor") (bitmap-type
"wxBITMAP_TYPE_CUR")
  (x 10) (y 10)))
```

## 6.12. wxDatabase is-a wxObject

See also *Database classes overview* (page 205)

Not yet implemented.
----------------------

Every database object represents an ODBC connection. The connection may be closed and reopened.

### wxDatabase close

**bool** (close)

Resets the statement handles of any associated recordset objects, and disconnects from the current data source.

### wxDatabase create

**long** (database-create)

Creates a new ODBC database handle. The constructor of the first wxDatabase instance of an application initializes the ODBC manager.

### wxDatabase delete

**bool** (delete)

Destructor. Resets and destroys any associated wxRecordSet instances.

The destructor of the last wxDatabase instance will deinitialize the ODBC manager.

### wxDatabase error-occurred

**bool** (error-occurred)

Returns 1 if the last action caused an error.

**wxDatabase get-database-name****string (get-database-name)**

Returns the name of the database associated with the current connection.

**wxDatabase get-data-source****string (get-data-source)**

Returns the name of the connected data source.

**wxDatabase get-error-code****string (wxDatabase get-error-code)**

Returns the error code of the last ODBC function call. This will be a string containing one of:

SQL\_ERROR    General error.  
SQL\_INVALID\_HANDLE    An invalid handle was passed to an ODBC function.  
SQL\_NEED\_DATA    ODBC expected some data.  
SQL\_NO\_DATA\_FOUND    No data was found by this ODBC call.  
SQL\_SUCCESS    The call was successful.  
SQL\_SUCCESS\_WITH\_INFO    The call was successful, but further information can be obtained from the ODBC manager.

**wxDatabase get-error-message****string (get-error-message)**

Returns the last error message returned by the ODBC manager.

**wxDatabase get-error-number****long (get-error-number)**

Returns the last native error. A native error is an ODBC driver dependent error number.

**wxDatabase is-open****bool (is-open)**

Returns 1 if a connection is open.

**wxDatabase open**

**bool** (**open string** *datasource* **optional long** *exclusive* = 1 **optional string** *readonly* = 1 **optional string** *username* = "ODBC" **optional string** *password* = "")

Connect to a data source. *datasource* contains the name of the ODBC data source. The parameters *exclusive* and *readonly* are not used.

### 6.13. wxDate is-a wxObject

A class for manipulating dates.

Not yet implemented.
----------------------

#### **wxDate add-months**

**bool** (**add-months long** *months*)

Adds the given number of months to the date, returning TRUE if successful.

#### **wxDate add-weeks**

**bool** (**add-weeks long** *weeks*)

Adds the given number of weeks to the date, returning TRUE if successful.

#### **wxDate add-years**

**bool** (**add-years long** *years*)

Adds the given number of months to the date, returning TRUE if successful.

#### **wxDate create**

**void** (**create**)

Constructs a date object, initialized to zero. You are responsible for deleting this object when you have finished with it.

**void** (**create long** *month long day long year*)

Constructs a date object with the specified date. You are responsible for deleting this object when you have finished with it.

*month* is a number from 1 to 12.

*day* is a number from 1 to 31.

*year* is a year, such as 1995, 2005.



**wxDate create-julian****bool** (**create-julian** long *julian*)

Constructor taking an integer representing the Julian date.

**wxDate create-string****bool** (**wxDate create-string** string *date*)

Constructor taking a string representing a date. This must be either the string TODAY, or of the form MM/DD/YYYY or MM-DD-YYYY. For example:

```
(make-instance (gensym*) (date-string "11/26/1966"))
```

**wxDate format****string** (**format**)

Formats the date into a string according to the current display type.

**wxDate get-day****long** (**get-day**)

Returns the numeric day (in the range 1 to 365).

**wxDate get-day-of-week****long** (**get-day-of-week**)

Returns the integer day of the week (in the range 1 to 7).

**wxDate get-day-of-week-name****string** (**day-of-week-name**)

Returns the name of the day of week.

**wxDate get-day-of-year****long** (**get-day-of-year**)

Returns the day of the year (from 1 to 365).

**wxDate get-days-in-month**

**long (get-days-in-month)**

Returns the number of days in the month (in the range 1 to 31).

**wxDate get-first-day-of-month****long (get-first-day-of-month)**

Returns the day of week that is first in the month (in the range 1 to 7).

**wxDate get-julian-date****long (get-julian-date)**

Returns the Julian date.

**wxDate get-month****long (get-month)**

Returns the month number (in the range 1 to 12).

**wxDate get-month-end****long (get-month-end)**

Returns a new date representing the day that is last in the month. The new date must be deleted when it is finished with.

**wxDate get-month-name****string (get-month-name)**

Returns the name of the month.

**wxDate get-month-start****wxDate (get-month-start)**

Returns a new date representing the first day of the month. The new date must be deleted when it is finished with.

**wxDate get-week-of-month****long (get-week-of-month)**

Returns the week of month (in the range 1 to 6).

**wxDate get-week-of-year****long (get-week-of-year)**

Returns the week of year (in the range 1 to 52).

**wxDate get-year****long (get-year)**

Returns the year as an integer (such as '1995').

**wxDate get-year-end****wxDate (get-year-end)**

Returns a new date the date representing the last day of the year. Delete the new date when you have finished with it.

**wxDate get-year-start****wxDate (get-year-start)**

Returns a new date the date representing the first day of the year. Delete the new date when you have finished with it.

**wxDate is-leap-year****bool (is-leap-year)**

Returns TRUE if the year of this date is a leap year.

**wxDate set****bool (set)**

Sets the date to current system date.

**wxDate set-julian****bool (set-julian long *julian*)**

Sets the date to the given Julian date.

**wxDate set-date**

**bool (set-date long month long day long year)**

Sets the date to the given date.

*month* is a number from 1 to 12.

*day* is a number from 1 to 31.

*year* is a year, such as 1995, 2005.

### **wxDatE set-format**

**bool (set-format string format)**

Sets the current format type.

*format* should be one of:

wxDAY	Format day only.
wxMONTH	Format month only.
wxMDY	Format MONTH, DAY, YEAR.
wxFULL	Format day, month and year in US style: DAYOFWEEK, MONTH, DAY, YEAR.
wxEUROPEAN	Format day, month and year in European style: DAY, MONTH, YEAR.

### **wxDatE set-option**

**bool (set-option string option long enable=1)**

Enables or disables an option for formatting. *option* may be one of:

wxNO_CENTURY	The century is not formatted.
wxDATE_ABBR	Month and day names are abbreviated to 3 characters when formatting.

### **wxDatE add-days**

**wxDatE (add-days long days)**

Adds an integer number of days to the date, returning a new date object.

### **wxDatE subtract-days**

**wxDatE (subtract-days long days)**

Subtracts an integer number of days from the date, returning a new date object.

### **wxDatE subtract**

**long (subtract long *date1* long *date2*)**

Subtracts one date from another, return the number of intervening days.

**wxDat add-self**

**bool (add-self long *days*)**

Adds an integer number of days to the date, returning TRUE if successful.

**wxDat subtract-self**

**bool (subtract-self long *days*)**

Subtracts an integer number of days from the date, returning TRUE if successful.

**wxDat le**

**bool (le long *date*)**

Compare two dates, returning TRUE if the current date object is earlier than *date*.

**wxDat leq**

**bool (leq long *date*)**

Function to compare two dates, returning TRUE if the current date object is earlier or equal to *date*.

**wxDat ge**

**bool (ge long *date*)**

Function to compare two dates, returning TRUE if the current date object is later than *date*.

**wxDat geq**

**dboollong (geq long *date*)**

Function to compare two dates, returning TRUE if the current date object is later than or equal to *date*.

**wxDat eq**

**bool (eq long *date*)**

Function to compare two dates, returning TRUE if the current date object is equal to *date*.

## **wxDate neq**

**bool** (**neq** long *date*)

Function to compare two dates, returning TRUE if the current date object is not equal to *date*.

## **6.14. wxDC is-a wxObject**

See also *Overview* (page 202)

A wxDC (device context) is an abstraction of a surface that can be drawn onto.

The following functions can be used with any device context identifier, with the exception of `blit` which must not be used with a PostScript device context, and `get-text-extent-width`, `get-text-extent-height` which do not function correctly on PostScript or metafile device contexts.

## **wxDC begin-drawing**

**bool** (**begin-drawing**)

Bracket a series of drawing primitives in `begin-drawing` and `end-drawing` to optimize drawing under Windows, and also if drawing to a panel or dialog box context, for which these calls are mandatory. The calls may be nested.

## **wxDC blit**

**bool** (**blit** double *dest-x* double *dest-y* double *width* double *height* wxDC *source-dc* double *source-x* double *source-y* string *logical-op* = "wxCOPY")

Block-copies the given area from a source device context to a destination device context (the current object). This operation is not available to PostScript and Windows Metafile destination device contexts.

The argument *logical-op* sets the current logical function for the canvas. This determines how a source pixel from the source device context combines with a destination pixel in the current device context. It will most commonly be "wxCOPY", which simply draws with the current source pixels.

The possible values and their meaning in terms of source and destination pixel values are as follows:

wxAND	src AND dst
wxAND_INVERT	(NOT src) AND dst
wxAND_REVERSE	src AND (NOT dst)
wxCLEAR	0
wxCOPY	src
wxEQUIV	(NOT src) XOR dst
wxINVERT	NOT dst
wxNAND	(NOT src) OR (NOT dst)
wxNOR	(NOT src) AND (NOT dst)
wxNO_OP	dst

<code>wxOR</code>	<code>src OR dst</code>
<code>wxOR_INVERT</code>	<code>(NOT src) OR dst</code>
<code>wxOR_REVERSE</code>	<code>src OR (NOT dst)</code>
<code>wxSET</code>	<code>1</code>
<code>wxSRC_INVERT</code>	<code>NOT src</code>
<code>wxXOR</code>	<code>src XOR dst</code>

The most commonly used is `wxCOPY`. The others combine the current colour and the background using a logical operation. `wxXOR` is commonly used for drawing rubber bands or moving outlines, since drawing twice reverts to the original colour.

## **wxDC clear**

**bool (wxDC clear)**

Clears the device context using the background colour.

## **wxDC destroy-clipping-region**

**bool (destroy-clipping-region)**

Destroys the current clipping region.

## **wxDC draw-ellipse**

**bool (draw-ellipse double x double y double width double height)**

Draws an ellipse. The outline and filling attributes are determined by the pen and brush settings respectively.

## **wxDC draw-line**

**bool (draw-line double x1 double y1 double x2 double y2)**

Draws a line between the given points.

## **wxDC draw-lines**

**bool (draw-lines multifold *list*)**

Draws lines between the given points. *list* is a multifold, which can be created by a call to `mv-append` and a list of arguments. The list must contain an even number of floating-point values, interpreted in pairs as the points determining the multiline.

## **wxDC draw-point**

**bool (dc-draw-point double x double y)**

Draws a point.

**wxDC draw-polygon****bool (draw-polygon multifield *list*)**

Draws a (possibly filled) polygon. *list* is a multifield, which can be created by a call to *mv-append* and a list of arguments. The list must contain an even number of floating-point values, interpreted in pairs as the points determining the polygon. The outline and filling attributes are determined by the pen and brush settings respectively.

**wxDC draw-rectangle****bool (draw-rectangle double *x* double *y* double *width* double *height*)**

Draws a rectangle. The outline and filling attributes are determined by the pen and brush settings respectively.

**wxDC draw-rounded-rectangle****bool (draw-rounded-rectangle double *x* double *y* double *width* double *height* double *radius*)**

Draws a rounded rectangle, with corners with a specified radius (optional). The outline and filling attributes are determined by the pen and brush settings respectively.

**wxDC draw-text****bool (dc-draw-text string *text* double *x* double *y*)**

Draw text at the given position, using the font set by *set-font* (page 44), and using the colours set by *set-text-foreground* (page 44) and *set-text-background* (page 44) respectively.

**wxDC draw-spline****bool (draw-spline multifield *list*)**

Draws a spline curve. *list* is a multifield, which can be created by a call to *mv-append* and a list of arguments. The list must contain an even number of floating-point values, interpreted in pairs as the points determining the spline shape.

**wxDC end-doc****bool (end-doc)**

Ends a document (such as a PostScript or Windows printer document).

**wxDC end-drawing**



**bool (end-drawing)**

Bracket a series of drawing primitives in `begin-drawing` and `end-drawing` to optimize drawing under Windows, and also if drawing to a panel or dialog box context, for which these calls are mandatory. The calls may be nested.

**wxDC end-page****bool (end-page)**

Ends a page.

**wxDC get-min-x****double (get-min-x)**

Returns the minimum X value drawn so far on the device context.

**wxDC get-min-y****double (get-min-y)**

Returns the minimum Y value drawn so far on the device context.

**wxDC get-max-x****double (get-max-x)**

Returns the maximum X value drawn so far on the device context.

**wxDC get-max-y****double (get-max-y)**

Returns the maximum Y value drawn so far on the device context.

**wxDC get-text-extent-height****double (get-text-extent-height string *text*)**

Returns the height of the text as drawn on this device context, in logical units.

**wxDC get-text-extent-width****double (get-text-extent-width string *text*)**

Returns the width of the text as drawn on this device context, in logical units.

**wxDC ok****bool (ok)**

Returns TRUE if the device context is OK (usually meaning, it has been initialised correctly), and FALSE otherwise.

**wxDC start-doc****bool (start-doc string message)**

Starts a document (such as a PostScript or Windows printer document) using the given string for any associated message box (the message is not in fact currently used).

**wxDC start-page****bool (start-page)**

Starts a page.

**wxDC set-background****bool (set-background long brush)**

Sets the background brush.

**wxDC set-background-mode****bool (set-background-mode string mode)**

Sets the mode for drawing text background.

*mode* may be wxSOLID (use the text background colour) or wxTRANSPARENT (do not fill the background).

**wxDC set-brush****bool (set-brush wxBrush brush)**

Sets the current brush for the device context. *brush* is a *wxBrush* (page 19) object, or nil t select any existing brush out of the device context.

**wxDC set-colourmap****bool (set-colourmap wxColourMap cmap)**

Sets the colourmap for the device context. If *cmap* is nil, the original colourmap is restored so that it is safe to delete the device context (or colourmap).

### **wxDC set-clipping-region**

**bool (set-clipping-region double x1 double y1 double x2 double y2)**

Sets a rectangular clipping region, outside which drawing operations have no effect.

### **wxDC set-font**

**bool (set-font long font)**

Sets the current font for the device context. *font* is a *wxFont* (page 47) object, or nil to select any existing font out of the device context.

### **wxDC set-logical-function**

**bool (set-logical-function string logical-function)**

Sets the current logical function for the device context. The logical function determines how pixels are changed by the drawing functions, and may be one of *wxCOPY*, *wxXOR*, *wxINVERT*, *wxOR\_REVERSE* and *wxAND\_REVERSE*.

### **wxDC set-pen**

**bool (set-pen long pen)**

Sets the current pen for the device context. *pen* is a *wxPen* (page 69) object, or nil to select any existing pen out of the device context.

### **wxDC set-text-foreground**

**bool (set-text-foreground string colour)**

Sets the colour for the text foreground, effective when *draw-text* (page 41) is used. *colour* is a capitalized name from the list defined in the *wxWindows* manual.

### **wxDC set-text-background**

**bool (set-text-background string colour)**

Sets the colour for the text background, effective when *draw-text* (page 41) is used. *colour* is a capitalized name from the list defined in the *wxWindows* manual.

## **6.15. wxDialogBox is-a wxPanel**

See also *Overview* (page 202)

A dialog box is essentially a *wxPanel* (page 66) with its own *wxFrame* (page 48), and therefore shares some functions and behaviour with both of these objects.

Any panel item can be created as a child of a dialog box, and also the dialog box can be created *modal*, so that the flow of program control halts until the dialog box is dismissed.

The following event handlers are valid for the panel class:

- on-command** Override this to intercept panel item commands (such as button presses). See *wxCommandEvent* (page 25) for a list of event types associated with *wxCommandEvent*.
- on-event** Called with a *wxMouseEvent* (page 63) identifier. This can only be guaranteed only when the dialog box is in user edit mode (to be implemented).
- on-paint** Called with no arguments when the dialog box receives a repaint event from the window manager.
- on-size** The function is called with the window width and height.

## **wxDialogBox modal**

### **bool modal**

Initialize to TRUE if the dialog box is to be modal, FALSE otherwise. The default is FALSE.

## **wxDialogBox create**

### **void (create)**

The following slots may be initialized if not loading from a resource.

- *parent*: should be a *wxFrame* or *wxDialogBox*.
- *title*: a title for the dialog box caption.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this dialog box.
- *modal*: TRUE if the dialog is to be modal, FALSE otherwise (the default).
- *style*: may be absent or a string style: see below.

The following slots should be initialized if loading from a resource (see *Resource overview* (page 215) for further details).

- *parent*: should be a *wxFrame*.
- *resource*: the string name of the resource.

The value of **style** can be a *bit list* of the following values:

- wxCAPTION** Puts a caption on the dialog box (under XView and Motif this is mandatory).
- wxSTAY\_ON\_TOP** Stay on top of other windows (Windows only).
- wxSYSTEM\_MENU** Display a system menu (mandatory under XView and Motif).
- wxTHICK\_FRAME** Display a thick frame around the window (mandatory under XView and Motif).

**wxVSCROLL** Give the dialog box a vertical scrollbar (XView only).  
**wxDEFAULT\_DIALOG\_STYLE** Equivalent to a combination of **wxCAPTION**, **wxSYSTEM\_MENU** and **wxTHICK\_FRAME**

The default value for *style* is **wxDEFAULT\_DIALOG\_STYLE**.

If *modal* is **TRUE**, when the *show* message is sent to the dialog box object, the flow of control will stop until the *show* message has been called again with a **FALSE** parameter. Otherwise, if *modal* is **FALSE**, flow of control will immediately return to the program when the dialog box has been shown.

### **wxDialogBox on-char-hook**

**bool** (on-char-hook **wxKeyEvent** *event*)

Under Windows only, all key strokes going to a dialog box or frame can be intercepted before being passed on for normal processing. This handler takes the event object, and should return **TRUE** to override further processing, or **FALSE** to do default processing. See also *wxKeyEvent* (page 54).

### **wxDialogBox on-close**

**bool** (on-close)

The function is called when the user dismisses the dialog box. If the handler returns **TRUE**, the window is automatically deleted (possibly terminating the application). A return value of **FALSE** forbids automatic deletion.

### **wxDialogBox on-paint**

**void** (on-paint)

Override this handler to respond to paint events (sent when the dialog box needs repainting). Normally, a dialog box's items repaint themselves, but for special purposes, you may wish to draw on the dialog box device context.

### **wxDialogBox on-size**

**void** (on-size **long** *width* **long** *height*)

The function is called with the dialog box width and height when the user resizes the frame.

## **6.16. wxEvent is-a wxObject**

**wxEvent** is an 'abstract class' from which other event classes, such as mouse, key and command events, are derived.

### **wxEvent get-event-type**

**string (get-event-type)**

Returns the event type.

**6.17. wxEvtHandler is-a wxObject**

wxEvtHandler is an 'abstract class' for classes which have event handlers, such as wxCanvas or wxFrame. This class has yet to be documented.

**6.18. wxFont is-a wxObject**

A font is an object that can be set for a *device context* (page 115) to determine the characteristics of text drawn with *draw-text* (page 41). It can also be used to set panel item fonts.

**wxFont point-size****long point-size**

The point size of the font. The default is 10.

**wxFont family****symbol family**

The family of the font. May be one of wxROMAN, wxSCRIPT, wxDECORATIVE, wxSWISS, wxMODERN. The default is wxSWISS.

**wxFont style****symbol style**

The style of the font. May be one of wxNORMAL, wxITALIC, wxSLANT. The default is wxNORMAL.

**wxFont weight****symbol weight**

The weight of the font. May be one of wxNORMAL, wxLIGHT, wxBOLD. The default is wxNORMAL.

**wxFont underlined****bool underlined**

Whether the font is underlined (Windows only). May be TRUE or FALSE. The default is FALSE.

**wxFont create**

**void (create)**

Creates a font for use in a device context. The following slots can be used to initialize the font.

- *point-size* gives the font point size.
- *family* may be one of wxROMAN, wxSCRIPT, wxDECORATIVE, wxSWISS, wxMODERN, wxDEFAULT.
- *style* may be one of wxNORMAL, wxITALIC, wxSLANT.
- *weight* may be one of wxBOLD, wxLIGHT, wxNORMAL.
- *underlined* may be 1 or 0.

**6.19. wxFrame is-a wxWindow**

A wxFrame is a window containing text, canvas or panel subwindows. It normally has decorations added by the window manager, such as a system menu, a thick frame, and resize handles. When a wxWindows or wxCLIPS application initializes, a top-level frame must be returned to the system for successful start-up. When a top-level frame and all its children are deleted, the application terminates.

Usually an application will need to override the on-close handler in case the window manager sends the application a close message. If the handler returns TRUE, the frame is deleted by the system (possibly terminating the application).

See *wxWindow* (page 88) for message handlers in addition to the ones documented here.

**wxFrame create****void (create)**

The following slots may be initialized.

- *parent*: should be a wxFrame.
- *title*: a title for the dialog box caption.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this dialog box.
- *style*: may be absent or a string style: see below.

The style parameter may be a combination of the following, using the bitwise 'or' operator.

wxICONIZE      Display the frame iconized (minimized) (Windows only).  
wxCAPTION      Puts a caption on the frame (under XView and Motif this is mandatory).  
wxDEFAULT\_FRAME    Defined as a combination of wxMINIMIZE\_BOX, wxMAXIMIZE\_BOX,  
                         wxTHICK\_FRAME, wxSYSTEM\_MENU and wxCAPTION.  
wxMDI\_CHILD    Specifies a Windows MDI (multiple document interface) child frame.  
wxMDI\_PARENT    Specifies a Windows MDI (multiple document interface) parent frame.  
wxMINIMIZE      Identical to **wxICONIZE**.  
wxMINIMIZE\_BOX    Displays a minimize box on the frame (Windows only).  
wxMAXIMIZE      Displays the frame maximized (Windows only).  
wxMAXIMIZE\_BOX    Displays a maximize box on the frame (Windows only).  
wxSDI            Specifies a normal SDI (single document interface) frame.

<code>wxSTAY_ON_TOP</code>	Stay on top of other windows (Windows only).
<code>wxSYSTEM_MENU</code>	Displays a system menu (mandatory under XView and Motif).
<code>wxTHICK_FRAME</code>	Displays a thick frame around the window (mandatory under XView and Motif).

The function *show* must be called before a new frame is visible.

### **wxFrame create-status-line**

**bool** (**create-status-line** optional **long** *n=1*)

Creates a status line at the bottom of the frame. Use *set-status-text* (page 49) to write to the status line.

*n* is a number from 1 to 5 for the number of status areas to create.

### **wxFrame iconize**

**bool** (**iconize** optional **bool** *minimize*)

Minimizes the frame if the second argument is TRUE or absent, restores the frame otherwise.

### **wxFrame set-menu-bar**

**bool** (**set-menu-bar** **wxMenuBar** *menu-bar*)

Associates a menu bar with the frame. See *wxMenuBar* (page 60). You should not call this more than once for any given frame, and you should also not delete the *wxMenuBar* object once it has been assigned to a frame. It will be deleted when the *wxFrame* object is deleted.

### **wxFrame set-icon**

**bool** (**set-icon** **wxIcon** *icon*)

Sets the icon of a frame. See *wxIcon* (page 53).

### **wxFrame set-status-text**

**bool** (**set-status-text** **string** *text*, optional **long** *i=0*)

Sets the text for the status line (previously created with *create-status-line* (page 49)).

*i* is a number from 0 to 4 for the number of the status area to write to.

### **wxFrame set-title**

**bool** (**set-title** **string** *text*)



Set the title of a frame.

### **wxFrame set-tool-bar**

**bool** (**set-tool-bar** long *toolbar*)

Tells the MDI frame to manage the subwindow as a toolbar. Use in Windows MDI mode *only*.

### **wxFrame on-activate**

**void** (**on-activate** bool *active*)

Called the frame is activated or deactivated. Under Windows, you may need to intercept this message and set the focus for a subwindow, or the subwindow may not receive character events. By default, wxWindows will set the focus for the first subwindow of a frame.

### **wxFrame on-char-hook**

**bool** (**on-char-hook** wxKeyEvent *event*)

Under Windows only, all key strokes going to a dialog box or frame can be intercepted before being passed on for normal processing. This handler takes the event object, and should return TRUE to override further processing, or FALSE to do default processing. See also *wxKeyEvent* (page 54).

### **wxFrame on-close**

**bool** (**on-close**)

The function is called when the user dismisses the frame. If the handler returns TRUE, the window is automatically deleted (possibly terminating the application). A return value of FALSE forbids automatic deletion.

### **wxFrame on-menu-command**

**void** (**on-menu-command** long *menu-item*)

Called with the menu item identifier. Test the menu item identifier and perform an appropriate action.

### **wxFrame on-menu-select**

**void** (**on-menu-select** long *menu-item*)

Called with a menu item identifier, when the cursor travels over the menu item (but the user does not click). Test the menu item identifier and perform an appropriate action.

## **wxFrame on-size**

**void** (**on-size** long *width* long *height*)

The function is called with the frame width and height when the user resizes the frame. The application should define appropriate subwindow resizing behaviour in this handler, if appropriate.

The default handler performs child window resizing behaviour if there is only one child window. Otherwise, it gives up.

## **6.20. wxHelpInstance is-a wxObject**

Not yet implemented.
----------------------

A 'help instance' is created to manage on-line help associated with one or more files. wxCLIPS supports both Windows Help under MS Windows, and wxHelp under all platforms.

Windows Help (.hlp) files may be created using a number of tools, such as Tex2RTF. wxHelp (.xlp) files can be created with a text editor or a tool such as Tex2RTF.

wxHelp is very limited in its capabilities and should only be used on platforms with no native help. Consider using HTML files instead (although you cannot currently access HTML files from your application).

## **wxHelpInstance native**

**bool** **native**

If TRUE, the native help system will be invoked (such as WinHelp under MS Windows). If FALSE, wxHelp will be invoked.

## **wxHelpInstance create**

**void** (**create**)

Creates a help instance. If *native* is TRUE, the native help system will be invoked (such as WinHelp under MS Windows). If FALSE, wxHelp will be invoked.

## **wxHelpInstance display-block**

**bool** (**display-block** long *blockId*)

Displays the help file at the given block identifier (system dependent).

## **wxHelpInstance display-contents**

**bool** (**display-contents** string *filename*)

Displays the contents of the help file currently loaded.

**wxHelpInstance display-section****bool** (**display-section** **long** *section*)

Displays the help file at the given section (system dependent).

**wxHelpInstance keyword-search****bool** (**keyword-search** **string** *keyword*)

Positions the help file at a section matching the given string.

**wxHelpInstance load-file****bool** (**load-file** **string** *filename*)

Attempts to load the given file into the help instance. Use a function like `display-contents` to display the file.

**6.21. wxGauge is-a wxItem**

A gauge is used for displaying a quantity, for example amount of processing done. It must be a child of a `wxPanel` or `wxDialogBox`.

**wxGauge value****long** *value*

The current value of the gauge. The default is 1.

**wxGauge range****long** *range*

The range of the gauge. The default is 100.

**wxGauge create****void** (**create**)

Creates a gauge item on the given panel. The following slots may be initialized.

- *parent*: should be a `wxPanel` or `wxDialogBox`.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this item.

- *label*: may be absent or a string name to label this item.
- *style*: a bit list of values (see below).
- *value*: TRUE or FALSE.
- *range*: indicates the maximum value of the gauge.

*style* is a *bit list* of the following:

wxGA\_HORIZONTAL    The item will be created as a horizontal gauge.

wxGA\_VERTICAL      The item will be created as a vertical gauge.

wxGA\_PROGRESSBAR    Under Windows 95, the item will be created as a horizontal progress bar.

### **wxGauge set-bezel-face**

**bool** (**set-bezel-face** long *width*)

Set the bezel parameter of the gauge (takes effect under Windows version only).

### **wxGauge set-shadow-width**

**bool** (**set-shadow-width** long *width*)

Set the shadow width of the gauge (takes effect under Windows version only).

## **6.22. wxGroupBox is-a wxItem**

A wxGroupBox is a box drawn around one or more controls. Available under Windows only.

### **wxGroupBox create**

**void** (**create**)

Creates a group box. The following slots may be initialized.

- *parent*: should be a wxPanel or wxDialogBox.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this item.
- *label*: may be absent or a string name to label this item.
- *style*: reserved for future use.

## **6.23. wxIcon is-a wxBitmap**

An icon is a small bitmap which can be used to decorate a minimized frame. There are platform-specific ways of creating an icon.

### **wxIcon height**

**long height**

Height of the icon in pixels.

**wxIcon width****long width**

Width of the icon in pixels.

**wxIcon create****void (create)**

Loads an icon from a file or resource. Under X, the argument must be the filename of a valid XBM (X bitmap) file. Under Windows, the argument must be a icon filename, or the name of an icon resource compiled into the current executable.

Use *wxFrame set-icon* (page 49) to set the icon of a frame.

Under X, the permitted icon types in the *bitmap-type* are:

- **wxBITMAP\_TYPE\_BMP** Load a Windows bitmap file (if `USE_IMAGE_LOADING_IN_X` is enabled in `wx_setup.h`).
- **wxBITMAP\_TYPE\_GIF** Load a GIF bitmap file (if `USE_IMAGE_LOADING_IN_X` is enabled in `wx_setup.h`).
- **wxBITMAP\_TYPE\_XBM** Load an X bitmap file.
- **wxBITMAP\_TYPE\_XPM** Load an XPM (colour pixmap) file. Only available if `USE_XPM_IN_X` is enabled in `wx_setup.h`.

Under Windows, the permitted types are:

- **wxBITMAP\_TYPE\_ICO** Load a cursor from a `.ico` icon file (only if `USE_RESOURCE_LOADING_IN_MSW` is enabled in `wx_setup.h`).
- **wxBITMAP\_TYPE\_ICO\_RESOURCE** Load a Windows resource (as specified in the `.rc` file).

Examples:

```
; Under X
(bind ?icon (make-instance (gensym*)
  (bitmap-type wxBITMAP_TYPE_XBM)
  (filename "icon.xbm"))))

; Under Windows
(bind ?icon (make-instance (gensym*)
  (bitmap-type wxBITMAP_TYPE_ICO)
  (filename "icon.ico"))))
```

**6.24. wxKeyEvent is-a wxEvent**

A key event identifier is passed to a window's `on-char` or `on-char-hook` handler. The key code,

position and state of shift/control/alt can be examined by calling the following functions.

### **wxKeyEvent alt-down**

**bool (alt-down)**

Returns TRUE if alt was pressed.

### **wxKeyEvent control-down**

**bool (control-down)**

Returns TRUE if control was pressed.

### **wxKeyEvent get-key-code**

**string (get-key-code)**

Returns a string corresponding to the internal wxWindows key code, such as "WXK\_BACK", "WXK\_F1" or "WXK\_RETURN".

### **wxKeyEvent position-x**

**double (position-x)**

Gets the x position of the mouse pointer at the moment the key was pressed.

### **wxKeyEvent position-y**

**double (event-position-y)**

Gets the y position of the mouse pointer at the moment the key was pressed.

### **wxKeyEvent shift-down**

**bool (shift-down)**

Returns TRUE if shift was pressed.

## **6.25. wxListBox is-a wxItem**

A wxListBox displays a choice of strings. It must be the child of a panel or dialog box. In a single-selection listbox, only one choice may be highlighted. In a multiple-selection listbox, several may be highlighted.

### **wxListBox values**

**multifield values**

List of string values for initializing the wxListBox item.

**wxListBox multiple****bool multiple**

Initialize to TRUE for a multi-selection listbox, FALSE for a single-selection listbox.

**wxListBox create****void (create)**

Creates a list box item on the given panel or dialog box. The following slots may be initialized.

- *parent*: should be a wxPanel or wxDialogBox.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this item.
- *label*: may be absent or a string name to label this item.
- *style*: see below.
- *values*: a multifield list of strings.
- *multiple*: TRUE for a multiple-selection listbox, FALSE otherwise.

*style* is a *bit list* of some of the following:

wxNEEDED\_SB Create scrollbars if needed.

wxALWAYS\_SB Create scrollbars immediately.

wxHSCROLL Create horizontal scrollbar if contents are too wide (Windows only).

**wxListBox append****bool (append string item optional string client-data)**

Append a string to the list box, with an optional client data string.

**wxListBox find-string****long (find-string string item)**

Find the string in the list box and return the integer position if found, -1 if not.

**wxListBox clear****bool (clear)**

Clear all strings from the list box.

### **wxListBox get-selection**

**long (get-selection)**

Get the position of the selection (for single-selection list boxes only).

### **wxListBox get-string-selection**

**string (get-string-selection)**

Get the selected string (for single-selection list boxes only).

### **wxListBox set-selection**

**bool (set-selection long *item-pos* bool *flag=TRUE*)**

Set a selection by item position.

If *flag* is TRUE, the item will be selected, otherwise it will be deselected (multiple-selection listboxes only).

### **wxListBox set-string-selection**

**bool (set-string-selection string *item*)**

Set a selection by string.

### **wxListBox number**

**long (number)**

Return the number of items in the list box.

### **wxListBox delete-item**

**bool (delete-item long *item-pos*)**

Delete an item in the list box.

### **wxListBox get-string**

**string (get-string long *item-pos*)**

Return the string at the given position.



**wxListBox get-first-selection****long (get-first-selection)**

Get the first selection position in a multi-selection list box (-1 for no more selections).

**wxListBox get-next-selection****long (get-next-selection)**

Get the next selection position in a multi-selection list box (-1 for no more selections).

**6.26. wxMemoryDC is-a wxCanvasDC**

A memory device context is used for drawing into, or copying from, a bitmap. See also the *wxBitmap* (page 18) object.

**wxMemoryDC create****void (create)**

Create a memory device context using the current display depth. No slots need to be initialized.

**wxMemoryDC select-object****bool (select-object wxBitmap bitmap)**

Makes this device context the drawing surface for the given bitmap (see *wxBitmap* (page 18)). Deleting the memory device context disassociates the bitmap, freeing it to be used with another memory device context. To draw a bitmap on a device context that supports bitmap drawing (i.e. not a Metafile or PostScript device context), using code like the following:

```
;;; Utility function for drawing a bitmap
(defun draw-bitmap (?dc ?bitmap ?x ?y)
  (bind ?mem-dc (make-instance (gensym*) of wxMemoryDC))
  (send ?mem-dc select-object ?bitmap)
  ; Blit the memory device context onto the destination device context
  (send ?dc blit ?x ?y (send ?bitmap get-width) (send ?bitmap get-
height)
    ?mem-dc 0.0 0.0)
  (send ?mem-dc delete)
)
```

If *bitmap* is nil, the existing bitmap (if any) will be selected out of the device context. This might be necessary if you wish to delete the bitmap before deleting the device context (for example, for reusing the same device context for different bitmaps).

**6.27. wxMenu is-a wxWindow**

The menu is used as a component of a *wxMenuBar* (page 60) or as a popup menu. For a menu

bar, create menus, append menu items (strings, separators or further menus), and finally append the menu to the menu bar.

A menu or menu bar string may contain an ampersand, which is taken to mean 'underline the next character and use it as the hotkey'. This gives the user the opportunity to use keystrokes to access menus and items.

## **wxMenu callback**

### **symbol callback**

This slot should be initialized if creating a popup menu. The name represents a function that will be called with the `wxMenu` instance and `wxCommandEvent` instance when the user selects an item. Use `wxCommandEvent get-selection` to retrieve the selected menu item id.

## **wxMenu create**

### **void (create)**

Create a menu and returns the menu's ID. The following slots may be initialized.

- *callback*: should be present if creating a popup menu (i.e. not a menubar menu). It will be called with the `wxMenu` instance and `wxCommandEvent` instance when the user selects an item. Use `wxCommandEvent get-selection` to retrieve the selected menu item id.

## **wxMenu append**

**bool (append long *item-id* string *item-string* optional wxMenu *submenu* optional string *help-string* optional bool *checkable*)**

Append a string or submenu to the menu, passing the integer ID by which the menu item will be referenced, a string to be displayed, an optional pullright menu, and an optional flag for specifying whether this menu item can be checked.

A help string can be supplied, in which case the string will be shown on the first field of the status line (if any) in the frame containing the menu bar, when the mouse pointer moves over the menu item.

## **wxMenu append-separator**

### **bool (append-separator)**

Append a menu separator.

## **wxMenu break**

### **bool (break)**

Inserts a column break into the menu.

### **wxMenu check**

**bool** (**check** **long** *item-id* **bool** *check*)

Check (*check* = *TRUE* or uncheck *check* = *FALSE* the given menu item. MS Windows only.

### **wxMenu enable**

**bool** (**enable** **long** *item-id* **bool** *enable*)

Enable (*enable* = *TRUE* or disable *enable* = *FALSE* the given menu item.

## **6.28. wxMenuBar is-a wxWindow**

A menu bar is a standard user interface element which places the main commands of an application along the top of a *wxFrame* (page 123).

The menu bar must be assigned to a frame using *wxFrame set-menu-bar* (page 49). Once this is done, the menu bar must not be deleted by the application: it will be deleted when the frame is deleted.

A menu or menu bar string may contain an ampersand, which is taken to mean 'underline the next character and use it as the hotkey'. This gives the user the opportunity to use keystrokes to access menus and items.

See also *wxMenu* (page 58).

### **wxMenuBar create**

**void** (**create**)

Creates a menu bar.

### **wxMenuBar append**

**bool** (**append** **long** *menu-id* **string** *title*)

Appends a menu to a menu bar.

### **wxMenuBar check**

**bool** (**check** **long** *item-id* **bool** *check*)

Checks (*check* = *TRUE*) or unchecks (*check* = *FALSE*) the given menu item. MS Windows only.

### **wxMenuBar checked**

**bool** (checked long *item-id*)

Returns TRUE if the menu item is checked, FALSE otherwise.

### **wxMenuBar enable**

**bool** (enable long *item-id* bool *enable*)

Enables (*enable* = TRUE) or disables (*enable* = FALSE) the given menu item.

## **6.29. wxMessage is-a wxItem**

A wxMessage is a simple piece of text, or a bitmap, on a panel or dialog box.

### **wxMessage bitmap**

**wxMessage bitmap**

The bitmap associated with a wxMessage, if being used as a bitmap message.

### **wxMessage create**

**long** (create)

Creates a label or bitmap message item on the given panel.

The following slots may be used in initializing a wxButton instance:

- *parent*: should be a wxPanel or wxDialogBox.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *window-name*: may be absent or a string name to identify this message.
- *label*: for a text label message, must be a string.
- *bitmap*: for a bitmap label message, must be a wxBitmap.

## **6.30. wxMetaFile is-a wxObject**

Not yet implemented.
----------------------

A metafile is the Windows vector format. Currently, the only way of creating a Windows metafile is to close a metafile device context, and the only valid operations are to delete the metafile and to place it on the clipboard.

These functions are only available under Windows.

### **6.30.1. Example**

Below is a example of metafile, metafile device context and clipboard use. Note the way the metafile dimensions are passed to the clipboard, making use of the device context's ability to

keep track of the maximum extent of drawing commands.

```
(bind ?dc (make-instance (gensym*) of wxMetaFileDC))
(if (send ?dc ok) then
  (
    ; Do some drawing
    (bind ?mf (send ?dc close))
    (if (neq ?mf nil) then
      ; Pass metafile to the clipboard
      (send ?md set-clipboard (send ?dc get-max-x) (send ?dc get-max-
y))
      (send ?mf delete)
    )
  )
)
(send ?dc delete)
```

### **wxMetaFile set-clipboard**

**bool (wxMetaFile set-clipboard long width long height)**

Places the metafile on the clipboard, returning TRUE for success and FALSE for failure.

The metafile should be deleted immediately after this operation.

### **6.31. wxMetaFileDC is-a wxDC**

A metafile device context is used for creating a metafile. The programmer should create the metafile device context, close it to return a metafile, delete the device context, use the metafile (the only valid thing to do with it currently is to place it on the clipboard, and then delete the metafile.

These functions are only available under Windows.

See also *wxMetaFile* (page 61).

### **wxMetaFileDC filename**

**string filename**

Filename if creating this metafile device context as disk-based.

### **wxMetafileDC create**

**void (create)**

Creates a metafile device context.

*filename* is the file to be used if creating a disk-based metafile. Usually this will be zero or absent, and an in-memory metafile will be created.

**wxMetaFileDC close****wxMetaFile (close)**

Closes the metafile device context and returns a metafile (or nil if the function failed). The device context should no longer be used after this call is made, and it should be deleted.

See *wxMetaFile* (page 61).

**6.32. wxMouseEvent is-a wxEvent**

A mouse event identifier is passed to the canvas on-event handler. The state of the mouse buttons (and some keys) can be examined by calling the following functions.

**wxMouseEvent button****bool (button long *button*)**

Returns TRUE if the given button is changing state. *button* may be 1, 2 or 3 (left, middle and right buttons respectively).

**wxMouseEvent button-down****bool (button-down)**

Returns TRUE if the event is a mouse button down event.

**wxMouseEvent control-down****bool (control-down)**

Returns TRUE if the control key is down.

**wxMouseEvent dragging****bool (dragging)**

Returns TRUE if the event is a dragging event (holding a mouse button down and moving).

**wxMouseEvent left-down****bool (left-down)**

Returns TRUE if the left mouse button is down.

**wxMouseEvent left-up****bool (left-up)**

Returns TRUE if the left mouse button is up.

### **wxMouseEvent is-button**

**bool (is-button)**

Returns TRUE the event is a button press or release.

### **wxMouseEvent middle-down**

**bool (middle-down)**

Returns TRUE if the middle mouse button is down.

### **wxMouseEvent middle-up**

**bool (middle-up)**

Returns TRUE if the middle mouse button is up.

### **wxMouseEvent position-x**

**double (position-x)**

Returns the mouse x-position.

### **wxMouseEvent position-y**

**double (position-y)**

Returns the mouse y-position.

### **wxMouseEvent right-down**

**bool (right-down)**

Returns TRUE if the right mouse button is down.

### **wxMouseEvent right-up**

**bool (right-up)**

Returns TRUE if the right mouse button is up.

### **wxMouseEvent shift-down**

**bool (shift-down)**

Returns TRUE if the shift key is down.

**6.33. wxMultiText is-a wxText**

A multi-line text item is able to show several lines of text, unlike the single line *wxText* (page 81) item. It must be the child of a panel or dialog box.

**wxMultiText create****long (create)**

Creates a multi-line text item on the given panel or dialog box. The following slots may be initialized.

- *parent*: should be a *wxPanel* or *wxDialogBox*.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this item.
- *label*: may be absent or a string name to label this item.
- *style*: see below.
- *value*: a string for initializing the value of the multi-text.

The *style* parameter can be a *bit list* of the following:

**wxHSCROLL**    A horizontal scrollbar will be displayed. If **wxHSCROLL** is omitted, only a vertical scrollbar is displayed, and lines will be wrapped. This parameter is ignored under XView.

**wxREADONLY**    The text is read-only (not XView).

**6.34. wxObject**

*wxObject* is an 'abstract class' from which other *wxCool* classes are derived.

**wxObject dont-create****symbol dont-create**

Set on creation when it is not desirable for the usual underlying object creation to occur. Specifically, used when creating objects to wrap *wxCLIPS* integer identifiers for panel items created when loading in a dialog or panel resource. See *wx\_item.clp*, *wxPanel* handler *create-child-objects*.

**wxObject id****long id**



The integer identifier of the underlying wxCLIPS object.

### **wxObject pending-delete**

#### **bool pending-delete**

TRUE if the object is about to be deleted (an internal setting to avoid double deletion).

### **wxObject add-event-handlers**

#### **void (add-event-handlers)**

All classes should override (but still call) this handler in order to add callbacks for this instance. The wxObject version adds an OnDelete callback that will be called for all instances.

### **wxObject create**

#### **void (create)**

For wxObject, this is a no-operation that must be redefined by derived classes to perform per-instance initialization.

### **wxObject init after**

#### **void (init after)**

This handler is implemented to call the *create* handler after the slot initialization phase is complete. *create* is also defined for wxObject, as a no-operation, and must be redefined by each major subclass to do the construction for the instance.

## **6.35. wxPanel is-a wxCanvas**

A panel is a subwindow for placing panel items, such as the *wxButton* (page 20) and *wxText* (page 81) item. Its parent must be a *wxFrame* (page 48). A panel inherits most properties from canvas, except for scrollbar functionality.

Note that a *wxDialogBox* (page 44) may be used in a similar way to a panel.

The following event handlers are valid for the panel class:

- on-default-action** Override this to intercept double clicks in listboxes.
- on-command** Override this to intercept panel item commands (such as button presses). See *wxCommandEvent* (page 25) for a list of event types associated with *wxCommandEvent*.
- on-event** Called with a *wxMouseEvent* (page 63) identifier. This can only be guaranteed only when the panel is in user edit mode (to be implemented).
- on-paint** Called with no arguments when the panel receives a repaint event from the window manager.
- on-size** The function is called with the window width and height.

## **wxPanel resource**

### **string resource**

The name of the resource the panel or dialog is to be loaded from, if any. Initially the empty string.

## **wxPanel create**

### **void (create)**

Creates a panel.

The following slots may be initialized if not loading from a resource.

- *parent*: should be a `wxFrame`.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this pane;.
- *style*: may be absent or a string style: see below.

The following slots should be initialized if loading from a resource (see *Resource overview* (page 215) for further details).

- *parent*: should be a `wxFrame`.
- *resource*: the string name of the resource.

The *style* parameter may be a combination of the following, using the bitwise 'or' operator.

`wxABSOLUTE_POSITIONING` A hint to the windowing system not to try native Windowing system layout (Motif only). This is the recommended style for all Motif panels and dialog boxes.

`wxBORDER` Draws a thin border around the panel.

`wxVSCROLL` Gives the dialog box a vertical scrollbar (XView only).

### **void (on-default-action wxItem item)**

A panel receives this message in response to a double click in a listbox.

## **wxPanel on-command**

### **void (on-command wxItem item wxCommandEvent command-event)**

A panel or dialog box receives this message in response to a command (such as a button press), if the item has not overridden on-command. See *wxCommandEvent* (page 25) for a list of event types associated with `wxCommandEvent`.

**wxPanel set-button-font****bool** (**set-button-font** **wxFont** *font*)

Sets the font used for panel or dialog box item buttons (or contents). See also *set-label-font* (page 68).

**wxPanel set-label-font****bool** (**set-label-font** **wxFont** *font*)

Sets the font used for panel or dialog box item labels. See also *set-button-font* (page 68).

**wxPanel set-label-position****bool** (**set-label-position** **string** *position*)

Change the current label orientation for panel items: *position* may be *wxVERTICAL* or *wxHORIZONTAL*.

**wxPanel new-line****bool** (**new-line**)

Insert a new line, that is, make subsequent panel items appear at the start of the next line.

**6.36. wxItem is-a wxWindow**

A panel item is a control (or widget) that can be placed on a *wxPanel* (page 66) or *wxDialogBox* (page 44) to accept user input, and display information.

The following functions apply to panel items, which include *wxButton* (page 20), *wxCheckbox* (page 23), *wxChoice* (page 23), *wxMessage* (page 61), *wxText* (page 81), *wxMultiText* (page 65), *wxSlider* (page 80).

**wxItem get-label****string** (**get-label**)

Get the item's label.

**wxItem on-command****void** (**on-command** **wxItem** *item* **wxCommandEvent** *command-event*)

An item receives this message in response to a command (such as a button press). If this handler is not overridden, then *on-command* is sent to the item's parent panel. It is usually more convenient to override this handler for a panel rather than per panel item.

See *wxcommandevent* (page 25) for a list of event types associated with `wxCommandEvent`.

### **wxItem set-default**

**bool** (**set-default**)

Make this item the default.

### **wxItem set-label**

**bool** (**set-label** string *label*)

Set the item's label.

## **6.37. wxPen is-a wxObject**

A pen is used to control the colour and style of subsequent drawing operations on a *device context* (page 115).

### **wxPen colour**

**string** colour

The colour for initializing the `wxPen`.

### **wxPen style**

**symbol** style

The style for initializing the `wxPen`. May be one of `wxSOLID`, `wxDOT`, `wxLONG_DASH`, `wxSHORT_DASH`, `wxTRANSPARENT`.

### **wxPen create**

**void** (**create**)

Creates a pen for use in a device context. A pen is used for the outlines of graphic shapes. A brush must be set to fill the shapes.

The following slots may be initialized.

- *colour* is a `wxWindows` colour string such as "BLACK", "CYAN".
- *width* specifies the width of the pen.
- *style* may be one of `wxSOLID`, `wxDOT`, `wxLONG_DASH`, `wxSHORT_DASH`, `wxTRANSPARENT`.

## **6.38. wxPostScriptDC is-a wxDC**

A `wxPostScriptDC` is used for drawing into a postscript file.

**wxPostScriptDC filename****string filename**

The filename associated with the device context.

**wxPostScriptDC interactive****bool interactive**

TRUE if the creation of the device context should pop up a printer dialog.

**wxPostScriptDC window****wxWindow window**

Initialize this to the parent window for any dialogs the device context will pop up. Defaults to nil.

**wxPostScriptDC create****void (create)**

Creates a postscript device context. The following slots may be initialized.

- *filename* is the file to be used for printing to.
- *interactive* may be TRUE to popup up a printer dialog, or FALSE otherwise.
- *window* is a parent window for the printer dialog.

**6.39. wxPrinterDC is-a wxDC**

A wxPrinterDC is used for drawing onto a Windows printer.

**wxPrinterDC device****string device**

The device name for this device context. Defaults to the empty string.

**wxPrinterDC driver****string driver**

The driver name for this device context. Defaults to the empty string.

**wxPrinterDC filename****string filename**

The filename associated with the device context, if printing to a file.

### **wxPrinterDC interactive**

#### **bool interactive**

TRUE if the creation of the device context should pop up a printer dialog.

### **wxPrinterDC window**

#### **wxWindow window**

Initialize this to the parent window for any dialogs the device context will pop up. Defaults to nil.

### **wxPrinter create**

#### **void (create)**

Creates a printer device context. The following slots may be initialized.

- *device* is the Windows device name (defaults to the empty string).
- *driver* is the Windows printer driver name (defaults to the empty string).
- *filename* is the file to be used for printing to.
- *interactive* may be TRUE to pop up a printer dialog, or FALSE otherwise.

## **6.40. wxRadioBox is-a wxItem**

A radiobox item is a matrix of strings with associated radio buttons. The buttons are mutually exclusive, so pressing one will deselect the current selection.

### **wxRadioBox major-dimension**

#### **long major-dimension**

Specifies the number of rows (if style is wxVERTICAL) or columns (if style is wxHORIZONTAL) for a two-dimensional radiobox.

### **wxRadioBox values**

#### **multifield values**

List of string values for initializing the wxRadioBox labels.

### **wxRadioBox create**

#### **void (create)**

Creates a radiobox item on the given panel or dialog box. The following slots may be initialized.

- *parent*: should be a `wxPanel` or `wxDialogBox`.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this item.
- *label*: may be absent or a string name to label this item.
- *style*: A bit list of values (see below).
- *values*: a multifield list of strings for the radiobox labels.
- *major-dimension*: specifies the number of rows (if style is `wxVERTICAL`) or columns (if style is `wxHORIZONTAL`) for a two-dimensional radiobox.

*style* may be a *bit list* of:

`wxVERTICAL`    Lays the radiobox out in columns.  
`wxHORIZONTAL`    Lays the radiobox out in rows.

### **wxRadioBox get-selection**

**long (get-selection)**

Get the ID of the button currently selected.

### **wxRadioBox set-selection**

**bool (set-selection long item)**

Sets the given button to be the current selection.

## **6.41. wxRecordSet is-a wxObject**

See also *Database classes overview* (page 205)

Not yet implemented.
----------------------

Each recordset represents an ODBC database query. You can make multiple queries at a time by using multiple recordsets with a database or you can make your queries in sequential order using the same recordset.

### **wxRecordSet database**

**wxDatabase database**

The parent database.

### **wxRecordSet type**

**wxRecordSet type**

The initial type of the recordset. Currently there are two possible values of *type*:

- "wxOPEN\_TYPE\_DYNASET": Loads only one record at a time into memory. The other data of the result set will be loaded dynamically when moving the cursor. This is the default type.
- "wxOPEN\_TYPE\_SNAPSHOT": Loads all records of a result set at once. This will need much more memory, but will result in faster access to the ODBC data.

## **wxRecordSet create**

### **void (create)**

Constructs a recordset object, and appends the recordset object to the parent database's list of recordset objects, for later destruction when the database is destroyed.

The following slots may be initialized.

- *database*: the parent wxDatabase.
- *type*: the type of recordset, see below.
- *options*: not yet used.

Currently there are two possible values of *type*:

- "wxOPEN\_TYPE\_DYNASET": Loads only one record at a time into memory. The other data of the result set will be loaded dynamically when moving the cursor. This is the default type.
- "wxOPEN\_TYPE\_SNAPSHOT": Loads all records of a result set at once. This will need much more memory, but will result in faster access to the ODBC data.

## **wxRecordSet delete**

### **bool (delete)**

Deletes the recordset. All data except that stored in user-defined variables will be lost. It also unlinks the recordset object from the parent database's list of recordset objects.

## **wxRecordSet execute-sql**

### **bool (execute-sql string sql)**

Directly executes a SQL statement. The data will be presented as a normal result set. Note that the recordset must have been created as a snapshot, not dynaset. Dynasets will be implemented in the near future.

Examples of common SQL statements are given in *A selection of SQL commands* (page 209).

## **wxRecordSet get-char-data**

### **string (get-char-data string-or-long col)**



Returns the character (string) data for the current record at the specified column. The column can be a name or an integer position (starting from zero).

### **wxRecordSet get-col-name**

**string** (**get-col-name** long *col*)

Gets the name of the column at position *col*. Returns the empty string if *col* does not exist.

### **wxRecordSet get-col-type**

**string** (**get-col-type** string-or-long *col*)

Gets the name of the column at position *col* or name *col*. Returns "SQL\_TYPE\_NULL" if *col* does not exist.

See *ODBC SQL data types* (page 208) for the possible return values from this function.

### **wxRecordSet get-columns**

**long** (**get-columns** optional string *table* = "")

Returns the columns of the table with the specified name. If no name is given, the internal class member *table* will be used. If both names are NULL nothing will happen. The data will be presented as a normal result set, organized as follows:

0 (VARCHAR) TABLE\_QUALIFIER

1 (VARCHAR) TABLE\_OWNER

2 (VARCHAR) TABLE\_NAME

3 (VARCHAR) COLUMN\_NAME

4 (SMALLINT) DATA\_TYPE

5 (VARCHAR) TYPE\_NAME

6 (INTEGER) PRECISION

7 (INTEGER) LENGTH

8 (SMALLINT) SCALE

9 (SMALLINT) RADIX

10 (SMALLINT) NULLABLE

11 (VARCHAR) REMARKS

**wxRecordSet get-data-sources****bool (get-data-sources)**

Gets the currently-defined data sources via the ODBC manager. The data will be presented as a normal result set. See the documentation for the ODBC function `SQLDataSources` for how the data is organized. The name of the source is at column 0.

**wxRecordSet get-error-code****string (get-error-code)**

Returns the error code of the last ODBC action. This will be a string containing one of:

`SQL_ERROR` General error.  
`SQL_INVALID_HANDLE` An invalid handle was passed to an ODBC function.  
`SQL_NEED_DATA` ODBC expected some data.  
`SQL_NO_DATA_FOUND` No data was found by this ODBC call.  
`SQL_SUCCESS` The call was successful.  
`SQL_SUCCESS_WITH_INFO` The call was successful, but further information can be obtained from the ODBC manager.

**wxRecordSet get-filter****string (get-filter)**

Returns the current filter.

**wxRecordSet get-float-data****double (get-float-data string-or-long col)**

Returns the floating-point data for the current record at the specified column. The column can be a name or an integer position (starting from zero).

**wxRecordSet get-foreign-keys****bool (get-foreign-keys optional string *ftable* = "" optional string *ktable* = "")**

Returns a list of foreign keys in the specified table (columns in the specified table that refer to primary keys in other tables), or a list of foreign keys in other tables that refer to the primary key in the specified table.

If *ptable* contains a table name, this function returns a result set containing the primary key of the specified table.

If *ftable* contains a table name, this functions returns a result set of containing all of the foreign keys in the specified table and the primary keys (in other tables) to which they refer.

If both *ptable* and *ftable* contain table names, this function returns the foreign keys in the table

specified in *f*table that refer to the primary key of the table specified in *p*table. This should be one key at most.

GetForeignKeys returns results as a standard result set. If the foreign keys associated with a primary key are requested, the result set is ordered by FKTABLE\_QUALIFIER, FKTABLE\_OWNER, FKTABLE\_NAME, and KEY\_SEQ. If the primary keys associated with a foreign key are requested, the result set is ordered by PKTABLE\_QUALIFIER, PKTABLE\_OWNER, PKTABLE\_NAME, and KEY\_SEQ. The following table lists the columns in the result set.

0 (VARCHAR)	PKTABLE_QUALIFIER
1 (VARCHAR)	PKTABLE_OWNER
2 (VARCHAR)	PKTABLE_NAME
3 (VARCHAR)	PKCOLUMN_NAME
4 (VARCHAR)	FKTABLE_QUALIFIER
5 (VARCHAR)	FKTABLE_OWNER
6 (VARCHAR)	FKTABLE_NAME
7 (VARCHAR)	FKCOLUMN_NAME
8 (SMALLINT)	KEY_SEQ
9 (SMALLINT)	UPDATE_RULE
10 (SMALLINT)	DELETE_RULE
11 (VARCHAR)	FK_NAME
12 (VARCHAR)	PK_NAME

### **wxRecordSet get-int-data**

**long (get-int-data string-or-long col)**

Returns the integer data for the current record at the specified column. The column can be a name or an integer position (starting from zero).

### **wxRecordSet get-number-cols**

**long (get-number-cols)**

Returns the number of columns in the result set.

### **wxRecordSet get-number-fields**

**long (get-number-fields)**

Not implemented.

### **wxRecordSet get-number-params**

**long (get-number-params)**

Not implemented.

**wxRecordSet get-number-records****long (get-number-records)**

Returns the number of records in the result set.

**wxRecordSet get-primary-keys****long (get-primary-keys optional string *table* = "")**

Returns the column names that comprise the primary key of the table with the specified name. If no name is given the class member *tablename* will be used. If both names are NULL nothing will happen. The data will be presented as a normal result set, organized as follows:

0 (VARCHAR)	TABLE_QUALIFIER
1 (VARCHAR)	TABLE_OWNER
2 (VARCHAR)	TABLE_NAME
3 (VARCHAR)	COLUMN_NAME
4 (SMALLINT)	KEY_SEQ
5 (VARCHAR)	PK_NAME

**wxRecordSet get-result-set****bool (get-result-set)**

Copies the data presented by ODBC into the recordset. Depending on the recordset type all or only one record(s) will be copied. Usually this function will be called automatically after each successful database operation.

**wxRecordSet get-table-name****string (get-table-name)**

Returns the name of the current table.

**wxRecordSet get-tables****bool (get-tables)**

Gets the tables of a database. The data will be presented as a normal result set, organized as follows:

0 (VARCHAR)	TABLE_QUALIFIER
1 (VARCHAR)	TABLE_OWNER
2 (VARCHAR)	TABLE_NAME
3 (VARCHAR)	TABLE_TYPE (TABLE, VIEW, SYSTEM TABLE, GLOBAL TEMPORARY, LOCAL TEMPORARY, ALIAS, SYNONYM, or database-specific type)
4 (VARCHAR)	REMARKS

**wxRecordSet goto****bool (goto long *n*)**

Moves the cursor to the record with the number *n*, where the first record has the number 0.

**wxRecordSet is-bof****TRUE (is-bof)**

Returns TRUE if the user tried to move the cursor before the first record in the set.

**wxRecordSet is-field-dirty****bool (is-field-dirty string-or-long *field*)**

Returns TRUE if the given field has been changed but not saved yet.

**wxRecordSet is-field-null****bool (is-field-null string-or-long *field*)**

Returns TRUE if the given field has no data.

**wxRecordSet is-col-nullable****bool (is-col-nullable string-or-long *field*)**

Returns TRUE if the given column may contain no data.

**wxRecordSet is-eof****bool (is-eof)**

Returns TRUE if the user tried to move the cursor behind the last record in the set.

**wxRecordSet is-open****bool (is-open)**

Returns TRUE if the parent database is open.

**wxRecordSet move****bool (move long *rows*)**

Moves the cursor a given number of rows. Negative values are allowed.

### **wxRecordSet move-first**

**bool** (move-first)

Moves the cursor to the first record.

### **wxRecordSet move-last**

**bool** (move-last)

Moves the cursor to the last record.

### **wxRecordSet move-next**

**bool** (move-next)

Moves the cursor to the next record.

### **wxRecordSet move-prev**

**bool** (move-prev)

Moves the cursor to the previous record.

### **wxRecordSet query**

**bool** (query string *columns* string *table* optional string *filter*)

Start a query. An SQL string of the following type will automatically be generated and executed: "SELECT columns FROM table WHERE filter".

### **wxRecordSet set-table-name**

**bool** (set-table-name string *table*)

## **Specify the name of the table you want to use. 6.42. wxServer is-a wxObject**

See also *Interprocess communication overview* (page 199)

A server object represents the server side of a DDE conversation.

### **wxServer service-name**

**string** service-name

The name of the service (or server).

### **wxServer create**

**void (create)**

Creates a server object, and returns an integer id if successful.

The *service-name* slot should be initialized with a string identifying this service to potential clients. Under UNIX, it should contain a valid port number.

**wxServer on-accept-connection****wxConnection (on-accept-connection string *topic*)**

Should be overridden to return an instance of the appropriate `wxConnection` class, or nil to reject the connection.

**6.43. wxSlider is-a wxItem**

A slider is a panel item for denoting a range of values. It must be a child of a panel or dialog box.

**wxSlider min****int min**

The slider minimum value.

**wxSlider max****int max**

The slider maximum value.

**wxSlider value****int value**

The value of the slider (set-value and get-value can be called after initialization).

**wxSlider create****long (create)**

Creates a horizontal slider item on the given panel or dialog box. The following slots may be initialized.

- *parent*: should be a `wxPanel` or `wxDialogBox`.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.

- *window-name*: may be absent or a string name to identify this item.
- *label*: may be absent or a string name to label this item.
- *style*: a bit list of values (see below).
- *value*: an integer for initializing the value of the slider.
- *min*: the minimum value (zero or greater).
- *max*: the maximum value (1 or greater).

*style* is a *bit list* of the following:

wxHORIZONTAL      The item will be created as a horizontal slider.

wxVERTICAL      The item will be created as a vertical slider.

## 6.44. wxText is-a wxItem

A text item is used for displaying and editing a single line of text. It must be a child of a panel or dialog box. See also *wxMultiText* (page 65) for multiline text items.

### wxText value

#### string value

The initial value. The handlers *put-value* and *get-value* are defined for this slot. *set-value* is a synonym for *put-value*.

### wxText create

#### long (create)

Creates a single-line text item on the given panel or dialog box. The following slots may be initialized.

- *parent*: should be a wxPanel or wxDialogBox.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this item.
- *label*: may be absent or a string name to label this item.
- *style*: see below.
- *value*: a string for initializing the value of the text item.

The *style* parameter can be a *bit list* of the following:

wxTE\_PROCESS\_ENTER      The callback function will receive the event  
                                 wxEVENT\_TYPE\_TEXT\_ENTER\_COMMAND. Note that this will break tab  
                                 traversal for this panel item under Windows. Single-line text only.

wxTE\_PASSWORD      The text will be echoed as asterisks. Single-line text only.

wxTE\_READONLY      The text will not be user-editable.

wxHSCROLL      A horizontal scrollbar will be displayed. If wxHSCROLL is omitted, only a vertical  
                         scrollbar is displayed, and lines will be wrapped. This parameter is ignored  
                         under XView. Multi-line text only.



**wxText set-value****bool** (**set-value** string *value*)

Set the text item's string value. A synonym for put-value.

**6.45. wxTextWindow is-a wxWindow**

To display a lot of text, use this subwindow as the child of a *wxFrame* (page 48). It is capable of loading and saving files of ASCII text, and the text can be edited directly.

The following callbacks are valid for the dialog box class:

**OnChar** (Not XView.) The function is called with the text window identifier, key code, and key event identifier. If the event is an ASCII keypress, the code will correspond to the ASCII code; otherwise, the programmer must refer to the constants defined in `common.h`, in the `wxWindows` library.

To invoke default processing, call `text-window-on-char` (to be implemented).

**OnSize** The function is called with the text window identifier, width and height.

**wxTextWindow clear****bool** (**clear**)

Clears the contents of a text subwindow. Returns TRUE if successful, FALSE otherwise.

**wxTextWindow copy****bool** (**copy**)

Copies the selected text to the clipboard.

**wxTextWindow cut****bool** (**cut**)

Copies the selected text to the clipboard, then removes the selection.

**wxTextWindow create****void** (**create**)

Creates a text subwindow. The following slots may be initialized.

- *parent*: should be a *wxFrame*.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.

- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this canvas.
- *style*: may be absent or a string style: see below.

*style* is a *bit list* of some of the following:

`wxBORDER`      Use this style to draw a thin border in Windows (non-native implementation only).  
`wxNATIVE_IMPL`      Use this style to allow editing under Windows, albeit with a 64K limitation.

### **wxTextWindow discard-edits**

**void (discard-edits)**

Discard any edits in the text window.

### **wxTextWindow get-contents**

**string (get-contents)**

Returns the contents (up to a maximum of 1000 characters).

### **wxTextWindow load-file**

**bool (load-file string filename)**

Load the file onto the text subwindow, returning TRUE for success, FALSE for failure.

### **wxTextWindow modified**

**bool (modified)**

Returns TRUE if the text has been modified, FALSE otherwise.

### **wxTextWindow paste**

**bool (paste)**

Pastes the text (if any) from the clipboard to the text window.

### **wxTextWindow save-file**

**bool (save-file string filename)**

Saves the text in the subwindow to the given file, returning TRUE for success, FALSE for failure.

### **wxTextWindow set-editable**

**Bool** (**set-editable** **bool** *editable*)

Sets the text window to be editable or read-only.

### **wxTextWindow write**

**bool** (**write string** *text*)

Writes the given string into the text window, at the current cursor point.

## **6.46. wxTimer is-a wxObject**

A timer object can be created to notify the application at regular intervals.

### **wxTimer create**

**void** (**create**)

Creates a timer object. Use timer-start to start the timer, and register a Notify callback function to receive notification.

### **wxTimer start**

**bool** (**start long** *milliseconds*)

Starts the timer, notifying at intervals of duration *milliseconds*.

### **wxTimer stop**

**bool** (**stop**)

Stops the timer.

## **6.47. wxToolBar is-a wxPanel**

See also *Overview* (page 203)

A toolbar is an array of bitmap buttons, implemented by drawing bitmaps onto a canvas, instead of using the native button implementation.

**Note:** under XView, wxToolBar inherits from wxCanvas, not wxPanel, due to limitations in the XView toolkit.

### **wxToolBar create-buttons**

**bool create-buttons**

Specify TRUE if the enhanced underlying wxButtonBar class is to be used (optimized for Windows), FALSE for the standard wxToolBar class. The default is TRUE.

**wxToolBar orientation****string orientation**

Specify wxVERTICAL for vertical layout, or wxHORIZONTAL for horizontal layout. Ignored if doing manual layout.

**wxToolBar rows-or-columns****long rows-or-columns**

The maximum number of rows or columns in this toolbar (depends on the value of *orientation*). Ignored if doing manual layout.

**wxToolBar add-separator****bool (add-separator long id)**

Adds a separator between tools: only functional under Windows 95, but harmless under other platforms.

**wxToolBar add-tool**

**bool (add-tool long id long index wxBitmap bitmap1 optional wxBitmap bitmap2 = nil optional bool is-toggle = FALSE optional double x = -1.0 optional double x = 1.0 optional long client-data = 0 optional string short-help-string="" optional string long-help-string="")**

Adds a tool to the toolbar. Pass at least one bitmap, the bitmap to be displayed when active and not depressed; and optionally, the bitmap to be displayed when the tool is depressed or toggled. Under Windows, only one bitmap is necessary, and under X, the second bitmap will be created automatically as the inverse of the first button if none is supplied.

You can specify whether the tool is allowed to toggle, and pass a position if you are not going to automatically layout the toolbar with *toolbar-layout*. You can associate client data with the tool.

*short-help-string* is only used by Windows 95 versions of wxCLIPS. The string is used to supply text for a tooltip, a small yellow window that appears as the mouse pointer hovers over the button.

*long-help-string* specifies a longer help string that can be used by the application.

**wxToolBar clear-tools****bool (clear-tools)**

Clears all the tools from the toolbar.

## **wxToolBar create**

### **void (create)**

Creates a toolbar. The following slots may be initialized.

- *parent*: should be a wxFrame.
- *x*: may be absent or -1 to denote default layout, or zero/positive integer.
- *y*: may be absent or -1 to denote default layout, or zero/positive integer.
- *width*: may be absent or -1 to denote default width, or a positive integer.
- *height*: may be absent or -1 to denote default height, or a positive integer.
- *window-name*: may be absent or a string name to identify this toolbar.
- *style*: may be absent or a string style: see below.
- *create-buttons*: should be 1 (the default) if the toolbar should superimpose the user-supplied buttons onto a larger 3D button. If 0, the tool will be the same size as the button, and the toggle state will be represented by inverting the tool (Windows) or adding a border (X).
- *orientation*: specify wxVERTICAL for vertical layout, or wxHORIZONTAL for horizontal layout. Ignored if doing manual layout.
- *rows-or-columns*: the maximum number of rows or columns in this toolbar (depends on the value of *orientation*). Ignored if doing manual layout.

*style* may be a *bit list* of:

- wxTB\_3DBUTTONS: gives a simple 3D look to the buttons.

Note that absolute tool positioning (or the layout function) does not work for buttonbars under Windows 95: instead, you can specify the number of rows for the toolbar, and use add-separator to achieve inter-tool spacing.

## **wxToolBar create-tools**

### **bool (create-tools)**

This should be called when creating Windows 95 buttonbars, after all tools have been added. It adds the tools to the toolbar. You can also call it for non-Windows 95 toolbars and buttonbars, in which case it will have no effect.

## **wxToolBar enable-tool**

### **bool (enable-tool long tool-id bool enable)**

Enables the tool (if *enable* is TRUE) or disables it (if *enable* is FALSE).

## **wxToolBar get-max-height**

### **double (get-max-height)**

Gets the maximum height of the toolbar when it has been automatically laid out.

### **wxToolBar get-max-width**

**double** (**get-max-width**)

Gets the maximum width of the toolbar when it has been automatically laid out.

### **wxToolBar get-tool-client-data**

**long** (**get-tool-client-data** **long** *tool-id*)

Returns the client data associated with the given tool.

### **wxToolBar get-tool-enabled**

**bool** (**get-tool-enabled** **long** *tool-id*)

Returns TRUE if the tool is enabled, FALSE otherwise.

### **wxToolBar get-tool-long-help**

**string** (**get-tool-long-help** **long** *tool-id*)

Returns the long help string.

### **wxToolBar get-tool-short-help**

**string** (**get-tool-short-help** **long** *tool-id*)

Returns the short help string.

### **wxToolBar get-tool-state**

**bool** (**get-tool-state** **long** *tool-id*)

Returns the tool state (TRUE for toggled on, FALSE for off).

### **wxToolBar layout**

**bool** (**layout**)

Lays out all the tools if automatic layout is required.

### **wxToolBar on-paint**

**void (on-paint)**

Calls the default toolbar paint handler. You may wish to call this if you override the default handler.

**wxToolBar set-default-size**

**bool** (**set-default-size** **long** *width* **long** *height*)

Sets the width and height of tool buttons (Windows only). The default is 24 by 22.

**wxToolBar set-margins**

**bool** (**set-margins** **long** *x* **long** *y*)

Sets the width and height of the toolbar margins and spacing, if automatic layout is being used.

**wxToolBar set-tool-long-help**

**bool** (**set-tool-long-help** **long** *tool-id* **string** *help-string*)

Sets the long help string for this tool.

**wxToolBar set-tool-short-help**

**bool** (**set-tool-short-help** **long** *tool-id* **string** *help-string*)

Sets the short help string for this tool.

**wxToolBar toggle-tool**

**bool** (**toggle-tool** **long** *tool-id* **bool** *toggle*)

Toggles the tool on or off.

**6.48. wxWindow is-a wxEvtHandler**

The `wxWindow` is an 'abstract' class, used to access the functionality of classes derived from it. Therefore, please refer to this section when considering other classes.

**wxWindow x**

**long** *x*

The *x* coordinate of the window.

**wxWindow y**

**long y**

The window y coordinate.

**wxWindow width****long width**

The window width.

**wxWindow height****long height**

The window height.

**wxWindow client-width****long client-width**

The window client width (space available for contents of this window).

**wxWindow client-height****long client-height**

The window client height (space available for contents of this window).

**wxWindow centre****bool (centre word *orientation*)**

*orientation* may be wxVERTICAL, wxHORIZONTAL or wxBOTH. Centres the window with respect to its parent (or desktop).

**wxWindow enable****bool (enable bool *enable*)**

If *enable* is TRUE, enables the window for input. If *enable* is FALSE, the window is disabled (greyed out in the case of a panel item).

**wxWindow find-window-by-name****wxWindow (find-window-by-name string *name*)**



Finds the descendant window for this window.

### **wxWindow find-window-by-label**

**wxWindow** (**find-window-by-label** string *label*)

Finds the descendant window for this window.

### **wxWindow fit**

**bool** (**fit**)

Fits the panel, dialog box or frame around its children.

### **wxWindow get-name**

**string** (**get-name**)

Gets the window's name (the 'name' parameter passed to a window constructor).

### **wxWindow get-parent**

**wxWindow** (**get-parent**)

Gets the window's parent, or nil if there no parent.

### **wxWindow make-modal**

**bool** (**make-modal** bool *modal*)

*modal* may be TRUE to disable all frames and dialog boxes except this one, or FALSE to enable all frames and dialogs again.

Has no effect under XView.

### **wxWindow popup-menu**

**bool** (**popup-menu** wxMenu *menu* double *x* double *y*)

Pops up a menu on the window, at the given position. The menu will be dismissed (but not destroyed) when the user makes a selection.

Note that there is a reliability problem with Motif popup menus; they may not pop up after the first time.

### **wxWindow set-cursor**

**bool (set-cursor wxCursor cursor)**

Sets the cursor for this window.

### **wxWindow set-focus**

**bool (set-focus)**

Set this window to have the keyboard focus.

### **wxWindow set-size**

**bool (set-size long x long y long width long height)**

Sets the position and size of the window.

### **wxWindow set-client-size**

**bool (set-client-size long width long height)**

Sets the client size (available space for child windows) of the window.

### **wxWindow show**

**bool (show long show)**

If *show* is *TRUE*, shows the window. If *show* is *FALSE*, the window is hidden. If the window is a modal dialog box, *show = TRUE* will start the modal loop, and *show = FALSE* will terminate the loop (allowing execution to proceed after the first call to *show*).

## 7. wxCLIPS function groups

This is the reference for CLIPS windowing and other, miscellaneous functions. With these functions, it is possible to create special-purpose user interfaces independent of platform. Currently these capabilities are supported under MS Windows, Open Look and Motif.

### 7.1. How to use this reference

In the function definitions below, bold words are types, and are not part of CLIPS syntax. Parameter names are in italics. Types are as follows:

- **double** is a double-precision floating point number.
- **long** is a long integer.
- **string** is a double-quoted ASCII string.
- **word** is an unquoted string.
- **multifield** is a CLIPS multi-field value list.

Parameters can be **optional**, in which case defaults are assumed.

Some parameters can be *bit lists* of flags. wxCLIPS mimics the compact C++ syntax by parsing strings, for example:

```
(frame-create ... "wxSDI | wxDEFAULT")
```

Each identifier in such a parameter is translated to an integer value, and all are logical-or'ed together to produce an integer which is passed to the appropriate wxWindows C++ function.

**Note:** In Windows NT or WIN32s versions of Hardy, integer identifiers can be negative. So when validating integer identifiers, test for values of zero or -1, rather than for values less than zero.

Functions are grouped by class: in the underlying C++ library wxWindows, these are actual C++ classes. The functions are used in an object-oriented way, in that long integer identifiers represent an object, or instance, of a particular class. Some functions operate on several classes of object; for example, the functions prefixed **window** operate on classes derived from window, such as canvas, frame, dialog box, panel item. Similarly, the functions prefixed **dc** operate on different kinds of device context.

Most functions either take an integer identifier (checking its type before doing the appropriate thing) or return a new one.

In C++, the application would derive new classes and override certain member functions, such as **OnClose**, to intercept messages or events sent to the window objects. In CLIPS, the same effect is achieved by registering callback functions for specific events, using *window-add-callback* (page 175).

### 7.2. Application

One object of class 'application' is always present, and its implementation depends upon the C++ application hosting the wxCLIPS environment.

If an application defines a function called app-on-init, the wxCLIPS user interface can start up the application from a standard menu item, or straightaway if the **-start** flag is used on the command line. This function is not relevant to embedded versions of wxCLIPS.

If `app-on-init` is defined, it must initialize the main frame and return its integer identifier, or zero if the application could not be initialized.

The following *callbacks* are valid for the `app` class.

**OnCharHook** Under Windows only, all key strokes going to a dialog box or frame can be intercepted before being passed on for normal processing. This callback function takes the window id and event id, and should return 1 to override further processing, or 0 to do default processing. If the function returns 0, the `OnCharHook` message will be sent to the active window. See also *Key event* (page 140).

## **app-create**

**long (app-create)**

Returns the identifier of the current application object. If called multiple times, will always return the same number since there is only one application object, which will have been created before `wxCLIPS` is initialized.

## **app-get-show-frame-on-init**

**long (app-get-show-frame-on-init long id)**

Returns 1 if the application will show the top-level frame automatically on initialization, 0 otherwise.

You can pass 0 or a return value from `app-create` for the *id* parameter.

## **app-on-init**

**long (app-on-init)**

If defined, should initialize the application and return the identifier of the top-level frame, or zero if there is no main window associated with the `CLIPS` program. If zero is returned, the `wxCLIPS` development window will be created if it does not already exist. Under Windows, you may call *show-ide-window* (page 187) from this function.

## **app-set-show-frame-on-init**

**void (app-set-show-frame-on-init long id long show)**

Called before `on-app-init` returns, can change the behaviour of `wxCLIPS` to not force a 'show' of the main frame. This might be needed if you wish to set the focus for a different window on initialization. *show* should be 0 to disable showing, 1 otherwise (the default behaviour).

You can pass 0 or a return value from `app-create` for the *id* parameter.

## **7.3. Bitmap**

A bitmap is a rectangular array of pixels, possibly in colour. A bitmap can be created in memory,

or loaded from an XBM file under X, or BMP file under Windows.

A bitmap can be drawn on a canvas by selecting it into a *memory-dc* (page 143) object and using *dc-blit* (page 115). Bitmaps can also be used to create buttons; see *button-create-from-bitmap* (page 96).

## **bitmap-create**

**long (bitmap-create float *width* float *height* optional int *depth*)**

Creates a bitmap in memory. The programmer can draw into the bitmap by selecting it into a memory device context, for later drawing on an output device context such as a canvas device context.

## **bitmap-delete**

**long (bitmap-delete long *bitmap-id*)**

Deletes the given bitmap.

## **bitmap-get-colourmap**

**long (bitmap-get-colourmap long *id*)**

Gets the colourmap associated with the bitmap; if none, zero will be returned.

## **bitmap-get-height**

**long (bitmap-get-height long *id*)**

Gets the height of the bitmap.

## **bitmap-get-width**

**long (bitmap-get-width long *id*)**

Gets the width of the bitmap.

## **bitmap-load-from-file**

**long (bitmap-load-from-file string *file* optional word *bitmap-type*)**

Loads a bitmap from a file, and returns a new bitmap identifier.

*bitmap-type* specifies the type of bitmap to be loaded, and may be one of:

- `wxBITMAP_TYPE_BMP`: Windows BMP (the default under Windows).
- `wxBITMAP_TYPE_XBM`: X monochrome bitmap (the default under X).

- `wxBITMAP_TYPE_GIF`: GIF bitmap (only under X).
- `wxBITMAP_TYPE_XPM`: XPM colour bitmap (under Windows and X if `wxCLIPS` has been compiled to include this option).
- `wxBITMAP_TYPE_RESOURCE`: Windows resource bitmap; unlikely to be used since the resources compiled into `wxCLIPS` cannot be changed from CLIPS.

Note that whether any of these formats are available depends on how `wxCLIPS` was compiled.

## 7.4. Brush

A brush is an object that can be set for a device context (see *canvas-get-dc* (page 97), *device context* (page 115)) and determines the fill colour and style for subsequent drawing operations.

See also *pen* (page 156).

### brush-create

**long (brush-create string colour word style)**

**long (brush-create long colour-value word style)**

Creates a brush for use in a device context.

*colour* is a `wxWindows` colour string such as "BLACK", "CYAN", and *style* may be one of `wxSOLID`, `wxTRANSPARENT`.

*colour-value* is a value returned from *colour-create* (page 103).

A brush must be set to fill graphic shapes.

### brush-delete

**long (brush-delete long brush-id)**

Deletes the given brush.

## 7.5. Button

A button is a rectangular control which can be placed on a *panel* (page 154) to invoke a function.

### button-create

**long (button-create long panel-id string callback string label  
optional long x optional long y  
optional long width optional long height optional string style optional string name)**

Creates a label button on the given panel. The callback may be the empty string ("") to denote no callback, or a word or string for the function name. The function will be called when the button is pressed, with the button ID as argument. If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

The *style* argument is reserved for future use.

*name* gives the button a name that can be retrieved with *window-get-name* (page 176).

### button-create-from-bitmap

**long (button-create-from-bitmap long *panel-id* string *callback* long *bitmap-id*  
optional long *x* optional long *y*  
optional long *width* optional long *height* optional string *style* optional string *name*)**

Creates a bitmap button on the given panel. The callback may be the empty string ("") to denote no callback, or a word or string for the function name. The function will be called when the button is pressed, with the button ID as argument. If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

The *style* argument is reserved for future use.

*name* gives the button a name that can be retrieved with *window-get-name* (page 176).

## 7.6. Canvas

A subwindow used for drawing arbitrary graphics. It must be the child of a *frame* (page 123).

The following callbacks are valid for the canvas class.

**OnChar** The function is called with the canvas identifier, key code, and key event identifier. If the event is an ASCII keypress, the code will correspond to the ASCII code; otherwise, the programmer must refer to the constants defined in `common.h`, in the `wxWindows` library. See also *Key event* (page 140).

**OnEvent** Called with a canvas identifier and a *mouse event* (page 148) identifier.

**OnScroll** Called with a canvas identifier and a *command event* (page 103) identifier.

**OnPaint** Called with a canvas identifier when the canvas receives a repaint event from the window manager.

**OnSize** The function is called with the window identifier, width and height.

See also *window-add-callback* (page 175).

### canvas-create

**long (canvas-create long *parent-id* optional long *x* optional long *y*  
optional long *width* optional long *height* optional string *style*="wxRETAINED" optional  
string *name*)**

Creates a canvas for drawing graphics on. *parent-id* must be a valid frame ID.

The value of **style** can be a *bit list* of the following values:

**wxBORDER** Gives the canvas a thin border (Windows 3 and Motif only).

**wxRETAINED** Gives the canvas a `wxWindows`-implemented backing store, making repainting much faster but at a potentially costly memory premium (XView and Motif only).

**wxBACKINGSTORE** Gives the canvas an X-implemented backing store (XView and Motif only). The X server may choose to ignore this request, whereas `wxRETAINED` is

always implemented under X.

*name* gives the canvas a name that can be retrieved with *window-get-name* (page 176).

### **canvas-get-dc**

**long** (**canvas-get-dc** long *canvas-id*)

Return the device context handle belonging to the canvas. The device context must be retrieved before anything can be drawn on the canvas. If your drawing function is parameterized by a device context, you will be able to pass other types of device context to your drawing routine, such as PostScript and Windows metafile device contexts.

### **canvas-get-scroll-page-x**

**long** (**canvas-get-scroll-page-x** long *canvas-id*)

Gets the number of lines per horizontal scroll page.

### **canvas-get-scroll-page-y**

**long** (**canvas-get-scroll-page-y** long *canvas-id*)

Gets the number of lines per vertical scroll page.

### **canvas-get-scroll-pos-x**

**long** (**canvas-get-scroll-pos-x** long *canvas-id*)

Gets the horizontal scroll position in scroll units.

### **canvas-get-scroll-pos-y**

**long** (**canvas-get-scroll-pos-y** long *canvas-id*)

Gets the vertical scroll position in scroll units.

### **canvas-get-scroll-range-x**

**long** (**canvas-get-scroll-range-x** long *canvas-id*)

Gets the number of horizontal scroll positions.

### **canvas-get-scroll-range-y**

**long** (**canvas-get-scroll-range-y** long *canvas-id*)



Gets the number of vertical scroll positions.

### **canvas-get-scroll-pixels-per-unit-x**

**long** (**canvas-get-scroll-pixels-per-unit-x** **long** *canvas-id*)

Gets the number of pixels per horizontal scroll unit, as set in *canvas-set-scrollbar* (page 98).

### **canvas-get-scroll-pixels-per-unit-y**

**long** (**canvas-get-scroll-pixels-per-unit-y** **long** *canvas-id*)

Gets the number of pixels per vertical scroll unit, as set in *canvas-set-scrollbar* (page 98).

### **canvas-on-char**

**long** (**canvas-on-char** **long** *panel-id* **long** *event-id*)

The default implementation of the OnChar callback. Call this to pass intercepted characters through to the canvas.

### **canvas-on-scroll**

**long** (**canvas-on-scroll** **long** *panel-id* **long** *event-id*)

The default implementation of the OnScroll callback.

### **canvas-set-scrollbar**

**long** (**canvas-set-scrollbar** **long** *canvas-id* **long** *x-unit-size* **long** *y-unit-size*  
**long** *x-length* **long** *y-length* **long** *x-page-length* **long** *y-page-length*)

Set the scrollbar for the given canvas. The first argument pair specifies the number of pixels per logical scroll unit, that is, the number of pixels to scroll when a scroll arrow is clicked. If either is zero or less, that scrollbar will not appear. The second pair specifies the size of the virtual canvas in logical scroll units. The third pair of arguments specify the number of scroll units per page, that is, the amount to scroll by when the scrollbar is page-scrolled (usually by clicking either side of the scrollbar handle).

### **canvas-set-scroll-page-x**

**void** (**canvas-set-scroll-page-x** **long** *canvas-id* **long** *value*)

Sets the number of lines per horizontal scroll page.

### **canvas-set-scroll-page-y**

**void (canvas-set-scroll-page-y long *canvas-id* long *value*)**

Sets the number of lines per vertical scroll page.

### **canvas-set-scroll-pos-x**

**void (canvas-set-scroll-pos-x long *canvas-id* long *value*)**

Sets the horizontal scroll position.

### **canvas-set-scroll-pos-y**

**void (canvas-set-scroll-pos-y long *canvas-id* long *value*)**

Sets the vertical scroll position.

### **canvas-set-scroll-range-x**

**void (canvas-set-scroll-range-x long *canvas-id* long *value*)**

Sets the number of positions on the horizontal scrollbar.

### **canvas-set-scroll-range-y**

**void (canvas-set-scroll-range-y long *canvas-id* long *value*)**

Sets the number of positions on the vertical scrollbar.

### **canvas-scroll**

**long (canvas-scroll long *canvas-id* long *x-position* long *y-position*)**

Scroll the canvas programmatically to the given scroll position. To convert from pixel position to scroll position, divide the pixel position by the scroll unit size you passed to *canvas-set-scrollbar*s (page 98).

### **canvas-view-start-x**

**long (canvas-view-start-x long *canvas-id*)**

Returns the first visible horizontal scroll position. Note this is in scroll units, not pixel, so to convert to pixel position you need to multiply this value by the result of *canvas-get-scroll-pixels-per-unit-x* (page 98).

### **canvas-view-start-y**

**long (canvas-view-start-y long *canvas-id*)**

Returns the first visible vertical scroll position. Note this is in scroll units, not pixel, so to convert to pixel position you need to multiply this value by the result of *canvas-get-scroll-pixels-per-unit-y* (page 98).

## 7.7. Checkbox

A checkbox is a small box with a label, and can be in one of two states. It must be the child of a *panel* (page 154).

### check-box-create

**long (check-box-create long *panel-id* string *callback* string *label*  
optional long *x* optional long *y*  
optional long *width* optional long *height* optional string *style*  
optional string *name*)**

Creates a checkbox on the given panel. The callback may be the empty string (""), to denote no callback, or a word or string for the function name. The function will be called when the checkbox is turned on or off, with the checkbox ID as argument. If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

The *style* parameter is reserved for future use.

*name* gives the checkbox a name that can be retrieved with *window-get-name* (page 176).

### check-box-set-value

**long (check-box-set-value long *check-box-id* long *value*)**

Set the check box value (0 or 1).

### check-box-get-value

**long (check-box-get-value long *check-box-id*)**

Gets the check box value (0 or 1).

## 7.8. Choice

A choice item is similar to a single-selection *listbox* (page 141) but normally only the current selection is displayed. It must be the child of a *panel* (page 154).

### choice-create

**long (choice-create long *panel-id* string *callback* string *label*  
optional long *x* optional long *y* optional long *width* optional long *height*  
optional multifield *strings* optional string *style* optional string *name*)**

Creates a choice item on the given panel. A choice consists of a list of strings, one of which may be selected and displayed at any one time. The callback may be the empty string (""), to denote no callback, or a word or string for the function name. The function will be called when an item in the choice list is selected, with the choice ID as argument. If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

*strings* should be a multifield of strings. Note that under Motif, it is recommended that the values are passed in this function, rather than using *choice-append*, because of the nature of Motif (i.e. horrible). Otherwise, things are likely to be messed up.

The *style* parameter is reserved for future use.

*name* gives the choice item a name that can be retrieved with *window-get-name* (page 176).

### **choice-append**

**long (choice-append long *choice-id* string *item*)**

Append the string *item* to the choice.

### **choice-find-string**

**long (choice-find-string long *choice-id* string *item*)**

Searches for the given string and if found, returns the position ID of the string.

### **choice-clear**

**long (choice-clear long *choice-id*)**

Clears all the strings from the choice item.

### **choice-get-selection**

**long (choice-get-selection long *choice-id*)**

Get the ID of the string currently selected.

### **choice-get-string-selection**

**string (choice-get-string-selection long *choice-id*)**

Get the string currently selected.

### **choice-set-selection**

**long (choice-set-selection long *choice-id* long *item-id*)**

Sets the choice selection to the given item ID (numbered from zero).

### **choice-set-string-selection**

**long (choice-set-string-selection long choice-id string item)**

Set the selection by passing the appropriate item string.

### **choice-get-string**

**string (choice-get-string long choice-id long item-id)**

Get the string associated with the given item ID.

### **choice-number**

**long (choice-number long choice-id)**

Returns the number of strings in the choice item.

## **7.9. Client**

See also *Interprocess communication overview* (page 199)

A client object represents the client side of a DDE conversation.

To delete a client object, use `object-delete`.

### **client-create**

**long (client-create)**

Creates a client object, returning the integer id of the object if successful. No event handlers need be defined for a client object.

A connection is not made until *client-make-connection* (page 102) is called.

### **client-make-connection**

**long (client-make-connection long id string host string service string topic)**

Makes a connection to a server, returning the id of the connection if successful.

*id* is the client id returned from `client-create`.

*host* is ignored under Windows, and should contain a valid internet host name under X.

*service* is a DDE service identifier (under X should contain a socket identifier).

*topic* is a topic name for this connection.

Any connection event handlers should be defined by the application code after this function is called, assuming the return result is not zero.

### 7.10. Colour

A colour value is not in fact a class, but a long integer which contains the values of the red, green and blue components in a colour. A colour value may be passed to pen and brush creation functions, and also to some *Grid* (page 129) member functions.

#### colour-create

**long** (**colour-create** **long** *red* **long** *green* **long** *blue*)

**long** (**colour-create** **long** *parent-id* **string** *name*) A colour value may be created either by passing red, green and blue values, or by passing a colour name such as RED.

#### colour-red

**long** (**colour-red** **long** *colour*)

Returns the red component of the colour, a number between 0 and 255.

#### colour-green

**long** (**colour-green** **long** *colour*)

Returns the green component of the colour, a number between 0 and 255.

#### colour-blue

**long** (**colour-blue** **long** *colour*)

Returns the blue component of the colour, a number between 0 and 255.

### 7.11. Command event

A command event is associated with each panel item or menu callback. It is not passed to the callback, so must be retrieved within a callback using *panel-item-get-command-event* (page 155).

The command event types are as follows:

- **wxEVENT\_TYPE\_BUTTON\_COMMAND**
- **wxEVENT\_TYPE\_CHECKBOX\_COMMAND**
- **wxEVENT\_TYPE\_CHOICE\_COMMAND**
- **wxEVENT\_TYPE\_LISTBOX\_COMMAND**
- **wxEVENT\_TYPE\_TEXT\_COMMAND**
- **wxEVENT\_TYPE\_TEXT\_ENTER\_COMMAND**
- **wxEVENT\_TYPE\_MULTITEXT\_COMMAND**
- **wxEVENT\_TYPE\_MENU\_COMMAND**

- `wxEVENT_TYPE_SLIDER_COMMAND`
- `wxEVENT_TYPE_RADIOBOX_COMMAND`
- `wxEVENT_TYPE_SET_FOCUS`
- `wxEVENT_TYPE_KILL_FOCUS`

### **command-event-get-selection**

**long (command-event-get-selection long *id*)**

Returns the identifier selection corresponding to the selected item, for example a listbox or menu item.

### **command-event-is-selection**

**long (command-event-is-selection long *id*)**

Returns 1 if the event was a selection event, 0 otherwise.

## **7.12. Connection**

See also *Connection overview* (page 200)

A connection object *id* is used for initiating DDE commands and requests using functions such as `connection-execute`, and it also has event handlers associated with it to respond to commands from the other side of the connection.

### **connection-advise**

**long (connection-advise long *id* string *item* string *data*)**

Called by a server application to pass data to a client (for example, when a spreadsheet cell has been updated, and the client is interested in this value).

*item* is the name of the item, and *data* is a string representing the item's data.

Returns 1 if successful, 0 otherwise.

### **connection-create**

**long (connection-create)**

Creates a connection object. Note that if you use the server `OnAcceptConnection` callback, the object will be created for you. If you use `OnAcceptConnectionEx` then you must call `connection-create` yourself from within that callback.

### **connection-execute**

**long (connection-execute long *id* string *data*)**

Called by a client application to execute a command in the server. Note there is no item in this command.

*data* is a string representing the item's data.

Returns 1 if successful, 0 otherwise.

To get a result from a server, you need to call `connection-request` explicitly, since `connection-execute` doesn't return data.

### **connection-disconnect**

**long (connection-disconnect long *id*)**

Called by a client or server application to terminate this connection. After this call, the connection *id* is no longer valid.

Returns 1 if successful, 0 otherwise.

### **connection-poke**

**long (connection-poke long *id* string *item* string *data*)**

Called by a client application to poke data into the server.

*item* is the name of the item, and *data* is a string representing the item's data.

Returns 1 if successful, 0 otherwise.

### **connection-request**

**string (connection-request long *id* string *item*)**

Called by a client application to request data from a server.

*item* is the name of the requested data item.

Returns a string representing the data if successful, the empty string otherwise.

### **connection-start-advise**

**long (connection-start-advise long *id* string *item*)**

Called by a client application to indicate interest in a particular piece of data in a server. The client connection should then receive `OnAdvise` messages when the data is updated in the server.

*item* is the name of the data item of interest.

Returns 1 if the advise loop is allowed, 0 otherwise.



## connection-stop-advise

**long (connection-stop-advise long *id* string *item*)**

Called by a client application to indicate a termination of interest in a particular piece of data in a server.

*item* is the name of the data item of interest.

Returns 1 if successful, 0 otherwise.

## 7.13. Cursor

A cursor is a small bitmap used for representing the mouse pointer. It can be set for a particular subwindow, using *window-set-cursor*, as a cue for what operations are possible in this window at this point in time.

At present, it is only possible to create a cursor in wxCLIPS from a fixed range of cursor types.

## cursor-create

**long (cursor-create string *stock-cursor-name*)**

Creates a stock cursor. *stock-cursor-name* must be one of the following:

- wxCURSOR\_ARROW
- wxCURSOR\_BULLSEYE
- wxCURSOR\_CHAR
- wxCURSOR\_CROSS
- wxCURSOR\_HAND
- wxCURSOR\_IBEAM
- wxCURSOR\_LEFT\_BUTTON
- wxCURSOR\_MAGNIFIER
- wxCURSOR\_MIDDLE\_BUTTON
- wxCURSOR\_NO\_ENTRY
- wxCURSOR\_PAINT\_BRUSH
- wxCURSOR\_PENCIL
- wxCURSOR\_POINT\_LEFT
- wxCURSOR\_POINT\_RIGHT
- wxCURSOR\_QUESTION\_ARROW
- wxCURSOR\_RIGHT\_BUTTON
- wxCURSOR\_SIZENESW
- wxCURSOR\_SIZENS
- wxCURSOR\_SIZENWSE
- wxCURSOR\_SIZEWE
- wxCURSOR\_SIZING
- wxCURSOR\_SPRAYCAN
- wxCURSOR\_WAIT
- wxCURSOR\_WATCH
- wxCURSOR\_BLANK
- wxCURSOR\_CROSS\_REVERSE (X only)
- wxCURSOR\_DOUBLE\_ARROW (X only)

- `wxCURSOR_BASED_ARROW_UP` (X only)
- `wxCURSOR_BASED_ARROW_DOWN` (X only)

### **cursor-delete**

**long (cursor-delete long *cursor-id*)**

Deletes the given cursor.

### **cursor-load-from-file**

**long (cursor-load-from-file string *filename* word *bitmap-type* optional long *hotspot-x* optional long *hotspot-y*)**

Loads a cursor from a file.

*hotspot-x* and *hotspot-y* are currently only used under Windows when loading from an icon file, to specify the cursor hotspot relative to the top left of the image.

Under X, the permitted cursor types in *bitmap-type* are:

- **`wxBITMAP_TYPE_XBM`** Load an X bitmap file

Under Windows, the permitted types are:

- **`wxBITMAP_TYPE_CUR`** Load a cursor from a .cur cursor file (only if `USE_RESOURCE_LOADING_IN_MSW` is enabled in `wx_setup.h`).
- **`wxBITMAP_TYPE_CUR_RESOURCE`** Load a Windows resource (as specified in the .rc file).
- **`wxBITMAP_TYPE_ICO`** Load a cursor from a .ico icon file (only if `USE_RESOURCE_LOADING_IN_MSW` is enabled in `wx_setup.h`). Specify *hotSpotX* and *hotSpotY*.

## **7.14. Database**

See also *Database classes overview* (page 205)

Every database object represents an ODBC connection. The connection may be closed and reopened.

### **database-close**

**long (database-close long *id*)**

Resets the statement handles of any associated recordset objects, and disconnects from the current data source.

### **database-create**

**long (database-create)**

Creates a new ODBC database handle and returns an id. The constructor of the first wxDatabase instance of an application initializes the ODBC manager.

### **database-delete**

**long (database-delete long *id*)**

Destructor. Resets and destroys any associated wxRecordSet instances.

The destructor of the last wxDatabase instance will deinitialize the ODBC manager.

### **database-error-occurred**

**long (database-error-occurred long *id*)**

Returns 1 if the last action caused an error.

### **database-get-database-name**

**string (database-get-database-name long *id*)**

Returns the name of the database associated with the current connection.

### **database-get-data-source**

**string (database-get-data-source long *id*)**

Returns the name of the connected data source.

### **database-get-error-code**

**string (database-get-error-code long *id*)**

Returns the error code of the last ODBC function call. This will be a string containing one of:

SQL\_ERROR    General error.

SQL\_INVALID\_HANDLE    An invalid handle was passed to an ODBC function.

SQL\_NEED\_DATA    ODBC expected some data.

SQL\_NO\_DATA\_FOUND    No data was found by this ODBC call.

SQL\_SUCCESS The call was successful.

SQL\_SUCCESS\_WITH\_INFO    The call was successful, but further information can be obtained from the ODBC manager.

### **database-get-error-message**

**string (database-get-error-message long *id*)**

Returns the last error message returned by the ODBC manager.

### **database-get-error-number**

**long** (**database-get-error-number** **long** *id*)

Returns the last native error. A native error is an ODBC driver dependent error number.

### **database-is-open**

**long** (**database-is-open** **long** *id*)

Returns 1 if a connection is open.

### **database-open**

**long** (**database-open** **long** *id* **string** *datasource* **optional long** *exclusive* = 1 **optional string** *readonly* = 1 **optional string** *username* = "ODBC" **optional string** *password* = "")

Connect to a data source. *datasource* contains the name of the ODBC data source. The parameters *exclusive* and *readonly* are not used.

## **7.15. Date**

A class for manipulating dates.

### **date-add-months**

**long** (**date-add-months** **long** *date* **long** *months*)

Adds the given number of months to the date, returning 1 if successful.

### **date-add-weeks**

**long** (**date-add-weeks** **long** *date* **long** *weeks*)

Adds the given number of weeks to the date, returning 1 if successful.

### **date-add-years**

**long** (**date-add-years** **long** *date* **long** *years*)

Adds the given number of months to the date, returning 1 if successful.

### **date-create**

**long** (**date-create**)

Constructs a date object, initialized to zero. You are responsible for deleting this object when you have finished with it.

**long (date-create long *month* long *day* long *year*)**

Constructs a date object with the specified date. You are responsible for deleting this object when you have finished with it.

*month* is a number from 1 to 12.

*day* is a number from 1 to 31.

*year* is a year, such as 1995, 2005.

### **date-create-julian**

**long (date-create-julian long *julian*)**

Constructor taking an integer representing the Julian date.

### **date-create-string**

**long (date-create-string string *date*)**

Constructor taking a string representing a date. This must be either the string TODAY, or of the form MM/DD/YYYY or MM-DD-YYYY. For example:

```
(bind ?date (date-create-string "11/26/1966"))
```

### **date-delete**

**long (date-delete long *date*)**

Deletes the date object.

### **date-format**

**string (date-format long *date*)**

Formats the date into a string according to the current display type.

### **date-get-day**

**long (date-get-day long *date*)**

Returns the numeric day (in the range 1 to 365).

### **date-get-day-of-week**

**long (date-get-day-of-week long date)**

Returns the integer day of the week (in the range 1 to 7).

**date-get-day-of-week-name**

**string (get-day-of-week-name long date)**

Returns the name of the day of week.

**date-get-day-of-year**

**long (date-get-day-of-year long date)**

Returns the day of the year (from 1 to 365).

**date-get-days-in-month**

**long (date-get-days-in-month long date)**

Returns the number of days in the month (in the range 1 to 31).

**date-get-first-day-of-month**

**long (date-get-first-day-of-month long date)**

Returns the day of week that is first in the month (in the range 1 to 7).

**date-get-julian-date**

**long (date-get-julian-date long date)**

Returns the Julian date.

**date-get-month**

**long (date-get-month long date)**

Returns the month number (in the range 1 to 12).

**date-get-month-end**

**long (date-get-month-end long date)**

Returns a new date representing the day that is last in the month. The new date must be deleted when it is finished with.

**date-get-month-name****string** (**date-get-month-name long** *date*)

Returns the name of the month.

**date-get-month-start****long** (**date-get-month-start long** *date*)

Returns a new date representing the first day of the month. The new date must be deleted when it is finished with.

**date-get-week-of-month****long** (**date-get-week-of-month long** *date*)

Returns the week of month (in the range 1 to 6).

**date-get-week-of-year****long** (**date-get-week-of-year long** *date*)

Returns the week of year (in the range 1 to 52).

**date-get-year****long** (**date-get-year long** *date*)

Returns the year as an integer (such as '1995').

**date-get-year-end****long** (**date-get-year-end long** *date*)

Returns a new date the date representing the last day of the year. Delete the new date when you have finished with it.

**date-get-year-start****long** (**date-get-year-start long** *date*)

Returns a new date the date representing the first day of the year. Delete the new date when you have finished with it.

**date-is-leap-year****long (date-is-leap-year long date)**

Returns 1 if the year of this date is a leap year.

**date-set-current-date****long (date-set-current-date long date)**

Sets the date to current system date.

**date-set-julian****long (date-set-julian long date long julian)**

Sets the date to the given Julian date.

**date-set-date****long (date-set-date long date long month long day long year)**

Sets the date to the given date.

*month* is a number from 1 to 12.

*day* is a number from 1 to 31.

*year* is a year, such as 1995, 2005.

**date-set-format****long (date-set-format long date string format)**

Sets the current format type.

*format* should be one of:

wxDAY	Format day only.
wxMONTH	Format month only.
wxMDY	Format MONTH, DAY, YEAR.
wxFULL	Format day, month and year in US style: DAYOFWEEK, MONTH, DAY, YEAR.
wxEUROPEAN	Format day, month and year in European style: DAY, MONTH, YEAR.

**date-set-option****long (date-set-option long date string option long enable=1)**

Enables or disables an option for formatting. *option* may be one of:



wxNO\_CENTURY      The century is not formatted.

wxDATE\_ABBR      Month and day names are abbreviated to 3 characters when formatting.

### **date-add-days**

**long (date-add-days long *date* long *days*)**

Adds an integer number of days to the date, returning a new date object.

### **date-subtract-days**

**long (date-subtract-days long *date* long *days*)**

Subtracts an integer number of days from the date, returning a new date object.

### **date-subtract**

**long (date-subtract long *date* long *date1* long *date2*)**

Subtracts one date from another, return the number of intervening days.

### **date-add-self**

**long (date-add-self long *date* long *days*)**

Adds an integer number of days to the date, returning 1 if successful.

### **date-subtract-self**

**long (date-subtract-self long *date* long *days*)**

Subtracts an integer number of days from the date, returning 1 if successful.

### **date-le**

**long (date-le long *date1* long *date2*)**

Function to compare two dates, returning 1 if *date1* is earlier than *date2*.

### **date-leq**

**long (date-leq long *date1* long *date2*)**

Function to compare two dates, returning 1 if *date1* is earlier than or equal to *date2*.

**date-ge****long (date-ge long *date1* long *date2*)**

Function to compare two dates, returning 1 if *date1* is later than *date2*.

**date-geq****long (date-geq long *date1* long *date2*)**

Function to compare two dates, returning 1 if *date1* is later than or equal to *date2*.

**date-eq****long (date-eq long *date1* long *date2*)**

Function to compare two dates, returning 1 if *date1* is equal to *date2*.

**date-neq****long (date-neq long *date1* long *date2*)**

Function to compare two dates, returning 1 if *date1* is not equal to *date2*.

**7.16. Device context**

See also *Overview* (page 202)

A device context is an abstraction of a surface that can be drawn onto.

The following functions can be used with any device context identifier, with the exception of `dc-blit` which must not be used with a PostScript device context, and `dc-get-text-extent-width`, `dc-get-text-extent-height` which do not function correctly on PostScript or metafile device contexts.

**dc-begin-drawing****long (dc-begin-drawing long *id*)**

Bracket a series of drawing primitives in `dc-begin-drawing` and `dc-end-drawing` to optimize drawing under Windows, and also if drawing to a panel or dialog box context, for which these calls are mandatory. The calls may be nested.

**dc-blit****long (dc-blit long *dest-dc-id* double *dest-x* double *dest-y* double *width* double *height* long *source-dc-id* double *source-x* double *source-y* optional string *logical-op* = "wxCOPY")**

Block-copies the given area from a source device context to a destination device context. This

operation is not available to PostScript and Windows Metafile destination device contexts.

The argument *logical-op* sets the current logical function for the canvas. This determines how a source pixel from the source device context combines with a destination pixel in the current device context.

The possible values and their meaning in terms of source and destination pixel values are as follows:

wxAND	src AND dst
wxAND_INVERT	(NOT src) AND dst
wxAND_REVERSE	src AND (NOT dst)
wxCLEAR	0
wxCOPY	src
wxEQUIV	(NOT src) XOR dst
wxINVERT	NOT dst
wxNAND	(NOT src) OR (NOT dst)
wxNOR	(NOT src) AND (NOT dst)
wxNO_OP	dst
wxOR	src OR dst
wxOR_INVERT	(NOT src) OR dst
wxOR_REVERSE	src OR (NOT dst)
wxSET	1
wxSRC_INVERT	NOT src
wxXOR	src XOR dst

The default is wxCOPY, which simply draws with the current colour. The others combine the current colour and the background using a logical operation. wxXOR is commonly used for drawing rubber bands or moving outlines, since drawing twice reverts to the original colour.

## **dc-clear**

**long (dc-clear long *dc-id*)**

Clears the device context using the background colour.

## **dc-delete**

**long (dc-delete long *dc-id*)**

Deletes a device context that has been explicitly created (so not a canvas DC).

## **dc-destroy-clipping-region**

**long (dc-destroy-clipping-region long *dc-id*)**

Destroys the current clipping region.

## **dc-draw-ellipse**

**long (dc-draw-ellipse long *dc-id*)**

**double x double y double width double height)**

Draws an ellipse. The outline and filling attributes are determined by the pen and brush settings respectively.

### **dc-draw-line**

**long (dc-draw-line long dc-id  
double x1 double y1 double x2 double y2)**

Draws a line between the given points.

### **dc-draw-lines**

**long (dc-draw-lines long dc-id multifold list)**

Draws lines between the given points. *list* is a multifold, which can be created by a call to mv-append and a list of arguments. The list must contain an even number of floating-point values, interpreted in pairs as the points determining the multiline.

### **dc-draw-point**

**long (dc-draw-point long dc-id double x double y)**

Draws a point.

### **dc-draw-polygon**

**long (dc-draw-polygon long dc-id multifold list)**

Draws a (possibly filled) polygon. *list* is a multifold, which can be created by a call to mv-append and a list of arguments. The list must contain an even number of floating-point values, interpreted in pairs as the points determining the polygon. The outline and filling attributes are determined by the pen and brush settings respectively.

### **dc-draw-rectangle**

**long (dc-draw-rectangle long dc-id  
double x double y double width double height)**

Draws a rectangle. The outline and filling attributes are determined by the pen and brush settings respectively.

### **dc-draw-rounded-rectangle**

**long (dc-draw-rounded-rectangle long dc-id  
double x double y double width double height double radius)**

Draws a rounded rectangle, with corners with a specified radius (optional). The outline and filling

attributes are determined by the pen and brush settings respectively.

### **dc-draw-text**

**long** (**dc-draw-text** **long** *dc-id*  
**string** *text* **double** *x* **double** *y*)

Draw text at the given position, using the font set by *dc-set-font* (page 120), and using the colours set by *dc-set-text-foreground* (page 121) and *dc-set-text-background* (page 121) respectively.

### **dc-draw-spline**

**long** (**dc-draw-spline** **long** *dc-id* **multifield** *list*)

Draws a spline curve. *list* is a multifield, which can be created by a call to *mv-append* and a list of arguments. The list must contain an even number of floating-point values, interpreted in pairs as the points determining the spline shape.

### **dc-end-doc**

**long** (**dc-end-doc** **long** *dc-id*)

Ends a document (such as a PostScript or Windows printer document).

### **dc-end-drawing**

**long** (**dc-end-drawing** **long** *id*)

Bracket a series of drawing primitives in *dc-begin-drawing* and *dc-end-drawing* to optimize drawing under Windows, and also if drawing to a panel or dialog box context, for which these calls are mandatory. The calls may be nested.

### **dc-end-page**

**long** (**dc-end-page** **long** *dc-id*)

Ends a page.

### **dc-get-min-x**

**double** (**dc-get-min-x** **long** *dc-id*)

Returns the minimum X value drawn so far on the device context.

### **dc-get-min-y**

**double** (**dc-get-min-y** **long** *dc-id*)

Returns the minimum Y value drawn so far on the device context.

**dc-get-max-x**

**double** (**dc-get-max-x long** *dc-id*)

Returns the maximum X value drawn so far on the device context.

**dc-get-max-y**

**double** (**dc-get-max-y long** *dc-id*)

Returns the maximum Y value drawn so far on the device context.

**dc-get-text-extent-height**

**double** (**dc-get-text-extent-height long** *dc-id* **string** *text*)

Returns the height of the text as drawn on this device context, in logical units.

**dc-get-text-extent-width**

**double** (**dc-get-text-extent-width long** *dc-id* **string** *text*)

Returns the width of the text as drawn on this device context, in logical units.

**dc-ok**

**long** (**dc-ok long** *id*)

Returns 1 if the device context is OK (usually meaning, it has been initialised correctly), and 0 otherwise.

**dc-start-doc**

**long** (**dc-start-doc long** *dc-id* **string** *message*)

Starts a document (such as a PostScript or Windows printer document) using the given string for any associated message box (the message is not in fact currently used).

**dc-start-page**

**long** (**dc-start-page long** *dc-id*)

Starts a page.

**dc-set-background****long (dc-set-background long *dc-id* long *brush*)**

Sets the background brush.

**dc-set-background-mode****long (dc-set-background-mode long *dc-id* string *mode*)**

Sets the mode for drawing text background.

*mode* may be wxSOLID (use the text background colour) or wxTRANSPARENT (do not fill the background).

**dc-set-brush****long (dc-set-brush long *dc-id* long *brush-id*)**

Sets the current brush for the device context. *brush-id* is an ID returned from a call to *brush-create* (page 95), or zero to select any existing brush out of the device context.

**dc-set-colourmap****long (dc-set-colourmap long *dc-id* long *cmap-id*)**

Sets the colourmap for the device context. If *cmap-id* is zero, the original colourmap is restored so that it is safe to delete the device context (or colourmap).

**dc-set-clipping-region****long (dc-set-clipping-region long *dc-id*  
double *x1* double *y1* double *x2* double *y2*)**

Sets a rectangular clipping region, outside which drawing operations have no effect.

**dc-set-font****long (dc-set-font long *dc-id* long *font-id*)**

Sets the current font for the device context. *font-id* is an ID returned from a call to *font-create* (page 123), or zero to select any existing font out of the device context.

**dc-set-logical-function****long (dc-set-logical-function long *dc-id* string *logical-function*)**

Sets the current logical function for the device context. The logical function determines how pixels are changed by the drawing functions, and may be one of `wxCOPY`, `wxXOR`, `wxINVERT`, `wxOR_REVERSE` and `wxAND_REVERSE`.

### **dc-set-pen**

**long (dc-set-pen long *dc-id* long *pen-id*)**

Sets the current pen for the device context. *pen-id* is an ID returned from a call to *pen-create* (page 156), or zero to select any existing pen out of the device context.

### **dc-set-text-foreground**

**long (dc-set-text-foreground long *dc-id* string *colour*)**

Sets the colour for the text foreground, effective when *dc-draw-text* (page 118) is used. *colour* is a capitalized name from the list defined in the `wxWindows` manual.

### **dc-set-text-background**

**long (dc-set-text-background long *dc-id* string *colour*)**

Sets the colour for the text background, effective when *dc-draw-text* (page 118) is used. *colour* is a capitalized name from the list defined in the `wxWindows` manual.

## **7.17. Dialog box**

See also *Overview* (page 202)

A dialog box is essentially a *panel* (page 154) with its own *frame* (page 123), and therefore shares some functions and behaviour with both of these objects.

Any panel item can be created as a child of a dialog box, and also the dialog box can be created *modal*, so that the flow of program control halts until the dialog box is dismissed.

The following callbacks are valid for the dialog box class: see also those listed for panels.

**OnCharHook** Under Windows only, all key strokes going to a dialog box or frame can be intercepted before being passed on for normal processing. This callback function takes the window id and event id, and should return 1 to override further processing, or 0 to do default processing. See also *Key event* (page 140).

### **dialog-box-create**

**long (dialog-box-create long *parent-id* string *title*  
optional long *modal* optional long *x* optional long *y*  
optional long *width* optional long *height* optional string *style* optional string *name*)**

Creates a dialog box. *parent-id* can be zero or a valid dialog or frame ID; *title* should be a string for the dialog box's title. The value of *modal* may be 1 (when *window-show* (page 179) is called with an argument of 1, the dialog blocks until *window-show* is called with an argument of 0) or 0



(dialog is modeless, and window-show returns immediately).

The *window-show* (page 179) function must be called with argument 1 to make the dialog visible, and with argument 0 to undisplay the dialog (and to dismiss a modal dialog).

The *style* parameter may be a combination of the following, using the bitwise 'or' operator:

wxCAPTION     Puts a caption on the dialog box (under XView and Motif this is mandatory).  
wxSTAY\_ON\_TOP     Stay on top of other windows (Windows only).  
wxSYSTEM\_MENU     Display a system menu (mandatory under XView and Motif).  
wxTHICK\_FRAME     Display a thick frame around the window (mandatory under XView and Motif).  
wxVSCROLL     Give the dialog box a vertical scrollbar (XView only).

The default value for *style* is "wxCAPTION | wxSYSTEM\_MENU | wxTHICK\_FRAME".

*name* gives the dialog box a name that can be retrieved with *window-get-name* (page 176).

### **dialog-box-create-from-resource**

**long (dialog-box-create-from-resource long parent-id string resource-name)**

Creates a dialog box from the given wxWindows resource. The resource file containing this resource must first have been loaded with *load-resource-file* (page 185).

Panel items on a panel or dialog box that has been created from a resource, do not have conventional callbacks. Therefore you need to intercept the OnCommand event for the panel or dialog box and test the name and event of the item passed to this callback.

### **dialog-box-is-modal**

**long (dialog-box-is-modal long parent-id)**

Returns 1 if the dialog box is modal, 0 otherwise.

### **dialog-box-set-modal**

**long (dialog-box-set-modal long parent-id, long modal)**

Sets the dialog box to be modal or non-modal, before window-show is issued. Pass 1 or 0 to *modal*.

## **7.18. Event**

An event is an 'abstract class' from which other event classes, such as mouse, key and command events, are derived.

### **event-get-event-type**

**string (event-get-event-type long id)**

Returns the event type.

## 7.19. Font

A font is an object that can be set for a *device context* (page 115) to determine the characteristics of text drawn with *dc-draw-text* (page 118).

### font-create

**long (font-create long *point-size* word *family* word *style* word *weight* long *underlined* optional string*facename*)**

Creates a font for use in a device context.

*point-size* gives the font point size.

*family* may be one of wxROMAN, wxSCRIPT, wxDECORATIVE, wxSWISS, wxMODERN, wxDEFAULT.

*style* may be one of wxNORMAL, wxITALIC.

*weight* may be one of wxBOLD, wxLIGHT, wxNORMAL.

*underlined* may be 1 or 0.

*facename* is an optional font facename, for specifying the exact font face required.

### font-delete

**long (font-delete long *font-id*)**

Deletes the given font.

## 7.20. Frame

A frame is a window containing text, canvas or panel subwindows. It normally has decorations added by the window manager, such as a system menu, a thick frame, and resize handles. When a wxWindows or wxCLIPS application initializes, a top-level frame must be returned to the system for successful start-up. When a top-level frame and all its children are deleted, the application terminates.

Usually an application will need to register an OnClose handler in case the window manager sends the application a close message. If the handler returns 1, the frame is deleted by the system (possibly terminating the application).

The user can register the following callbacks:

**OnActivate** Called with a frame identifier and integer flag, when the frame is activated.

Under Windows, you may need to intercept this event and set the focus for a subwindow, or the subwindow may not receive character events. By default, wxWindows will set the focus for the first subwindow of a frame.

**OnCharHook** Under Windows only, all key strokes going to a dialog box or frame can be

intercepted before being passed on for normal processing. This callback function takes the window id and event id, and should return 1 to override further processing, or 0 to do default processing. See also *Key event* (page 140).

**OnClose** The function is called with the window identifier. If the callback returns 1 and the function was called by the window manager, the window is automatically deleted (possibly terminating the application). A return value of 0 forbids automatic deletion.

**OnMenuCommand** Called with a frame identifier and menu item identifier. Test the menu item identifier and perform an appropriate action.

**OnMenuSelect** Called with a frame identifier and menu item identifier, when the cursor travels over the menu item (but the user does not click). Test the menu item identifier and perform an appropriate action.

**OnSize** The function is called with the window identifier, width and height.

See also *window-add-callback* (page 175).

## frame-create

**long** (**frame-create** **long** *parent-id* **string** *title*  
**optional long** *x* **optional long** *y*  
**optional long** *width* **optional long** *height* **optional string** *style* **optional string** *name*)

Creates a frame. *parent-id* can be zero or a valid frame ID; *title* should be a string for the frame's title.

The style parameter may be a combination of the following, using the bitwise 'or' operator.

**wxICONIZE** Display the frame iconized (minimized) (Windows only).  
**wxCAPTION** Puts a caption on the frame (under XView and Motif this is mandatory).  
**wxDEFAULT\_FRAME** Defined as a combination of **wxMINIMIZE\_BOX**, **wxMAXIMIZE\_BOX**, **wxTHICK\_FRAME**, **wxSYSTEM\_MENU**, and **wxCAPTION**.  
**wxMDI\_CHILD** Specifies a Windows MDI (multiple document interface) child frame.  
**wxMDI\_PARENT** Specifies a Windows MDI (multiple document interface) parent frame.  
**wxMINIMIZE** Identical to **wxICONIZE**.  
**wxMINIMIZE\_BOX** Displays a minimize box on the frame (Windows only).  
**wxMAXIMIZE** Displays the frame maximized (Windows only).  
**wxMAXIMIZE\_BOX** Displays a maximize box on the frame (Windows only).  
**wxSDI** Specifies a normal SDI (single document interface) frame.  
**wxSTAY\_ON\_TOP** Stay on top of other windows (Windows only).  
**wxSYSTEM\_MENU** Displays a system menu (mandatory under XView and Motif).  
**wxTHICK\_FRAME** Displays a thick frame around the window (mandatory under XView and Motif).

*name* gives the frame a name that can be retrieved with *window-get-name* (page 176).

The function *window-show* must be called before a new frame is visible.

## frame-create-status-line

**long** (**frame-create-status-line** **long** *parent-id*, **optional long** *n=1*)

Creates a status line at the bottom of the frame. Use *frame-set-status-text* (page 125) to write to the status line.

*n* is a number from 1 to 5 for the number of status areas to create.

### **frame-iconize**

**long (frame-iconize long *frame-id* optional long *minimize*)**

Minimize the frame if the second argument is 1 or absent, restore the frame otherwise.

### **frame-is-iconized**

**long (frame-is-iconized long *frame-id*)**

Returns 1 if the frame is iconized (minimized), 0 otherwise.

### **frame-on-size**

**long (frame-on-size long *frame-id* long *width* long *height*)**

Performs default processing for the OnSize event. Can be called from within an OnSize callback.

### **frame-set-menu-bar**

**long (frame-set-menu-bar long *frame-id* long *menu-bar-id*)**

Associate a menu bar with the frame. See *menu bar* (page 145).

### **frame-set-tool-bar**

**long (frame-set-tool-bar long *frame-id* long *tool-bar*)**

Informs an MDI parent window that a panel or canvas should be treated as a toolbar, and sized accordingly. Windows only.

### **frame-set-icon**

**long (frame-set-icon long *frame-id* long *icon-id*)**

Set the icon of a frame. See *icon* (page 138).

### **frame-set-status-text**

**long (frame-set-status-text long *frame-id* string *text*, optional long *i=0*)**

Sets the text for the status line (previously created with *frame-create-status-line* (page 124)).

*i* is a number from 0 to 4 for the number of the status area to write to.

**frame-set-title****long (frame-set-title long *frame-id* string *text*)**

Set the title of a frame.

**7.21. Help**

A 'help instance' is created to manage on-line help associated with one or more files. wxCLIPS supports both Windows Help under MS Windows, and wxHelp under all platforms.

Windows Help (.hlp) files may be created using a number of tools, such as Tex2RTF. wxHelp (.xlp) files can be created with a text editor or a tool such as Tex2RTF.

wxHelp is very limited in its capabilities and should only be used on platforms with no native help. Consider using HTML files instead (although you cannot currently access HTML files from your application).

**help-create****long (help-create optional long *native* = 1)**

Creates a help instance. If *native* is 1, the native help system will be invoked (such as WinHelp under MS Windows). If 0, wxHelp will be invoked.

**help-delete****long (help-delete long *id*)**

Deletes the help instance.

**help-display-block****long (help-display-block long *id* long *blockId*)**

Displays the help file at the given block identifier (system dependent).

**help-display-contents****long (help-display-contents long *id* string *filename*)**

Displays the contents of the help file currently loaded.

**help-display-section****long (help-display-section long *id* long *section*)**

Displays the help file at the given section (system dependent).

### **help-keyword-search**

**long (help-keyword-search long *id* string *keyword*)**

Positions the help file at a section matching the given string.

### **help-load-file**

**long (help-load-file long *id* string *filename*)**

Attempts to load the given file into the help instance. Use a function like help-display-contents to display the file.

## **7.22. HWND functions**

This group of functions allows MS Windows programs to perform a few operations on another program's window.

### **hwnd-find**

**long (hwnd-find string *title*)**

Searches for a window with the given title, and returns the window handle. Returns zero if none was found.

### **hwnd-iconize**

**long (hwnd-iconize long *hwnd* optional long *iconize=1*)**

Iconizes or deiconizes the given window.

### **hwnd-move**

**long (hwnd-move long *hwnd* long *x* long *y* long *width* long *height* optional long *repaint=1*)**

Moves and resizes the given window.

### **hwnd-refresh**

**long (hwnd-refresh long *hwnd* optional long *erase-background=1*)**

Refreshes (invalidates) the given window.

### **hwnd-send-message**

**long (hwnd-send-message long *hwnd* long *msg* long *wparam* long *lparam*)**

Sends a Windows message to the window.

### **hwnd-show**

**long (hwnd-show long *hwnd* optional long *show=1*)**

Shows or hides the given window.

### **hwnd-quit**

**long (hwnd-quit long *hwnd*)**

Sends a WM\_CLOSE message to the given window.

## **7.23. Gauge**

A gauge is used for displaying a quantity, for example amount of processing done. It must be a child of a *panel* (page 154).

### **gauge-create**

**long (gauge-create long *panel-id* string *label*  
long *range* optional long *x* optional long *y*  
optional long *width* optional long *height* optional string *style* optional string *name*)**

Creates a gauge item on the given panel.

*range* indicates the maximum value of the gauge.

*style* is a *bit list* of the following:

wxGA_HORIZONTAL	The item will be created as a horizontal gauge
wxGA_VERTICAL	The item will be created as a vertical gauge.
wxGA_PROGRESSBAR	Under Windows 95, the item will be created as a horizontal progress bar.

*name* gives the gauge a name that can be retrieved with *window-get-name* (page 176).

If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

### **gauge-set-value**

**long (gauge-set-value long *gauge-id* long *value*)**

Set the value of a gauge item.

**gauge-set-bezel-face**

**long (gauge-set-bezel-face long gauge-id long width)**

Set the bezel parameter of the gauge (takes effect under Windows version only).

**gauge-set-shadow-width**

**long (gauge-set-shadow-width long gauge-id long width)**

Set the shadow width of the gauge (takes effect under Windows version only).

**7.24. Grid**

See also *Overview* (page 210)

A subwindow used for displaying grids (matrices/tables). A grid can contain text or bitmaps.

**Note:** this functionality is implemented in Windows only, for the time being.

The following callbacks are valid for the grid class.

**OnPaint** Called with a window identifier when the window receives a repaint event from the window manager.

**OnSize** The function is called with the window identifier, width and height.

**OnCellLeftClick** The function is called with the window identifier, row, column, x, y, control down flag, shift down flag.

**OnCellRightClick** The function is called with the window identifier, row, column, x, y, control down flag, shift down flag.

**OnCellChange** The function is called with the window identifier, row, column.

**OnChangeLabels** The function is called with the window identifier.

**OnChangeSelectionLabel** The function is called with the window identifier.

See also *window-add-callback* (page 175).

**grid-adjust-scrollbars**

**void (grid-adjust-scrollbars long grid-id)**

Adjusts the scrollbars to suit the size of the grid: this may cause one or both to be hidden. You may wish to call this from code which alters the number of rows or columns (or height/width of rows or columns), or from an *OnSize* callback.

**grid-append-cols**

**long (grid-append-cols long grid-id long n optional long update-labels=1)** Inserts *n* columns at the end of the table, updating the grid labels if *update-labels* is 1.

Returns 1 if successful, 0 otherwise.

**grid-append-rows**

**long (grid-append-rows long grid-id long n optional long update-labels=1)** Inserts *n* rows at



the end of the table, updating the grid labels if *update-labels* is 1.

Returns 1 if successful, 0 otherwise.

### **grid-clear-grid**

**void (grid-clear-grid long grid-id)**

Clears the grid (untested).

### **grid-create**

**long (grid-create long parent-id optional long x optional long y  
optional long width optional long height optional string style=0 optional string  
name="grid")**

Creates a grid window. You must also call `grid-create-grid` after you call this function.

*parent-id* must be a valid frame ID.

*name* gives the canvas a name that can be retrieved with *window-get-name* (page 176).

### **grid-create-grid**

**long (grid-create-grid long grid-id long rows long cols  
optional long default-width optional long default-height)**

Initializes the size of the grid. Returns 1 if successful, 0 otherwise.

### **grid-delete-cols**

**long (grid-delete-cols long grid-id long position long n optional long update-labels=1)**

Deletes *n* columns starting at *position*, updating the grid labels if *update-labels* is 1.

Returns 1 if successful, 0 otherwise.

### **grid-delete-rows**

**long (grid-delete-rows long grid-id long position long n optional long update-labels=1)**

Deletes *n* rows starting at *position*, updating the grid labels if *update-labels* is 1.

Returns 1 if successful, 0 otherwise.

### **grid-get-cell-alignment**

**string (grid-get-cell-alignment long grid-id long row long col)** Gets the cell text alignment at *row*, *col*. The return value is one of `wxLEFT`, `wxRIGHT`, `wxCENTRE`.

### **grid-get-cell-background-colour**

**long (grid-set-cell-background-colour long *grid-id* optional long *row*=-1 optional long *col*=-1)**  
Gets the cell background colour.

The return value is a colour value of the kind created using *colour-create*.

If *row* and *col* are zero or greater, the returned colour is that of an individual cell. If these values are -1 or absent, the colour is the global, default cell background colour.

### **grid-get-cell-bitmap**

**long (grid-get-cell-bitmap long *grid-id* long *row* long *col*)** Returns the bitmap associated with the cell at *row*, *col*. If none has been set, 0 will be returned. See also *grid-set-cell-bitmap* (page 134).

### **grid-get-cell-text-colour**

**long (grid-set-cell-text-colour long *grid-id* optional long *row*=-1 optional long *col*=-1)** Gets the cell text colour.

The return value is a colour value of the kind created using *colour-create*.

If *row* and *col* are zero or greater, the returned colour is that of an individual cell. If these values are -1 or absent, the colour is the global, default cell text colour.

### **grid-get-cell-value**

**string (grid-get-cell-value long *grid-id* long *row* long *col*)** Gets the cell value at *row*, *col*.

### **grid-get-column-width**

**long (grid-get-column-width long *grid-id* long *col*)** Gets the given column width in pixels.

### **grid-get-cursor-column**

**long (grid-get-cursor-column long *grid-id*)** Returns the column of the currently selected cell.

### **grid-get-cursor-row**

**long (grid-get-cursor-row long *grid-id*)** Returns the row of the currently selected cell.

### **grid-get-rows**

**long (grid-get-rows long *grid-id*)**

Returns the number of rows in the grid.

**grid-get-cols****long (grid-get-cols long *grid-id*)**

Returns the number of columns in the grid.

**grid-get-editable****long (grid-get-editable long *grid-id*)**

Returns 1 if the grid is editable (the text field is showing), 0 otherwise.

**grid-get-label-alignment****string (grid-get-label-alignment long *grid-id* string *orientation*)** Gets the column label or row label alignment.

If *orientation* is wxVERTICAL, the row label alignment is returned. If *orientation* is wxHORIZONTAL, the column label alignment is returned.

The return value is one of wxLEFT, wxRIGHT, wxCENTRE.

**grid-get-label-background-colour****long (grid-get-label-background-colour long *grid-id*)** Gets the label background colour.

The return value is a colour value of the kind created using *colour-create*.

**grid-get-label-size****long (grid-get-label-size long *grid-id* string *orientation*)** Gets the column label height or row label width in pixels. If *orientation* is wxVERTICAL, the row label width is returned. If *orientation* is wxHORIZONTAL, the column label height is returned.**grid-get-label-text-colour****long (grid-get-label-text-colour long *grid-id*)** Gets the label text colour.

The return value is a colour value of the kind created using *colour-create*.

**grid-get-label-value****string (grid-get-label-value long *grid-id* string *orientation* long *position*)** *position* is the label row or column position (starting from zero).

Gets a column label or row label value.

If *orientation* is wxVERTICAL, the row label alignment is set. If *orientation* is wxHORIZONTAL, the column label alignment is set.

### **grid-get-row-height**

**long (grid-get-row-height long *grid-id* long *row*)** Gets the given row height in pixels.

### **grid-get-scroll-pos-x**

**long (grid-get-scroll-pos-x long *grid-id*)**

Returns the current scroll position in the horizontal dimension (in scroll positions, not pixels).

### **grid-get-scroll-pos-y**

**long (grid-get-scroll-pos-y long *grid-id*)**

Returns the current scroll position in the vertical dimension (in scroll positions, not pixels).

### **grid-get-text-item**

**long (grid-get-text-item long *grid-id*)**

Returns the identifier of the text item which is used for editing cells. Use this to set the label of the text item, for example, in an OnChangeSelectionLabel callback, which is called when the user selects a different cell.

### **grid-insert-cols**

**long (grid-insert-cols long *grid-id* long *position* long *n* optional long *update-labels*=1)** Inserts *n* columns in front of *position*, updating the grid labels if *update-labels* is 1.

Returns 1 if successful, 0 otherwise.

### **grid-insert-rows**

**long (grid-insert-rows long *grid-id* long *position* long *n* optional long *update-labels*=1)** Inserts *n* rows in front of *position*, updating the grid labels if *update-labels* is 1.

Returns 1 if successful, 0 otherwise.

### **grid-on-activate**

**void (grid-on-activate long *grid-id* long *active*)**

Call this function from a frame OnActivate callback. This function causes focus to be given to the text field (if in editable mode).

**grid-on-paint****void (grid-on-paint long *grid-id*)**

Call this function if you override the on-paint event handler.

**grid-on-size****void (grid-on-size long *grid-id* long *w* long *h*)**

Call this function if you override the on-size event handler.

**grid-set-cell-alignment**

**void (grid-set-cell-alignment long *grid-id* string *alignment* long *row* long *col*)** Sets the cell text alignment at *row*, *col* to the given value. *alignment* should be one of wxLEFT, wxRIGHT, wxCENTRE.

**grid-set-cell-background-colour**

**void (grid-set-cell-background-colour long *grid-id* long *colour* optional long *row*=-1 optional long *col*=-1)** Sets the cell background colour.

*colour* should be a colour value created with *colour-create*.

If *row* and *col* are zero or greater, the colour will be associated with an individual cell. If these values are -1 or absent, the colour will refer to all cells.

**grid-set-cell-bitmap**

**void (grid-set-cell-bitmap long *grid-id* long *bitmap-id* long *row* long *col*)** Associates a bitmap with the cell at *row*, *col*. If this is called, the bitmap will be displayed instead of text. Since colourmaps are not used in drawing the bitmap, use low-colour bitmaps if possible (16 colours or less).

**grid-set-cell-text-colour**

**void (grid-set-cell-text-colour long *grid-id* long *colour* optional long *row*=-1 optional long *col*=-1)** Sets the cell text colour.

*colour* should be a colour value created with *colour-create*.

If *row* and *col* are zero or greater, the colour will be associated with an individual cell. If these values are -1 or absent, the colour will refer to all cells.

**grid-set-cell-text-font**

**void (grid-set-cell-text-font long *grid-id* long *font-id* optional long *row*=-1 optional long *col*=-**

1) Sets the cell text font.

*font* should be a valid font identifier.

If *row* and *col* are zero or greater, the font will be associated with an individual cell. If these values are -1 or absent, the font will refer to all cells.

### **grid-set-cell-value**

**void (grid-set-cell-value long *grid-id* string *value* long *row* long *col*)** Sets the cell at *row*, *col* to the given value.

### **grid-set-column-width**

**void (grid-set-column-width long *grid-id* long *col* long *width*)** Sets the given column width in pixels.

### **grid-set-divider-pen**

**void (grid-set-divider-pen long *grid-id* long *pen-id*)**

Sets the pen for drawing the cell divisions (light grey by default). If *pen-id* is 0, the divisions are switched off.

### **grid-set-editable**

**void (grid-set-editable long *grid-id* long *editable*)**

If *editable* is 1, displays the text field for entering cell values. If *editable* is 0, hides the text field.

### **grid-set-grid-cursor**

**void (grid-set-grid-cursor long *grid-id* long *row* long *column*)** Sets the selection to the given cell.

### **grid-set-label-alignment**

**void (grid-set-label-alignment long *grid-id* string *orientation* string *alignment*)** Sets the column label or row label alignment.

If *orientation* is wxVERTICAL, the row label alignment is set. If *orientation* is wxHORIZONTAL, the column label alignment is set.

*alignment* should be one of wxLEFT, wxRIGHT, wxCENTRE.

### **grid-set-label-background-colour**

**void (grid-set-label-background-colour long *grid-id* long *colour*)** Sets the label background colour.

*colour* should be a colour value created with *colour-create*.

### **grid-set-label-size**

**void (grid-set-label-size long *grid-id* string *orientation* long *size*)** Sets the column label height or row label width in pixels. If *orientation* is wxVERTICAL, the row label width is set. If *orientation* is wxHORIZONTAL, the column label height is set.

A value of zero switches off the label in the specified dimension.

### **grid-set-label-text-colour**

**void (grid-set-label-text-colour long *grid-id* long *colour*)** Sets the label text colour.

*colour* should be a colour value created with *colour-create*.

### **grid-set-label-text-font**

**void (grid-set-label-text-font long *grid-id* long *font-id*)** Sets the label text font.

### **grid-set-label-value**

**void (grid-set-label-value long *grid-id* string *orientation* string *value* long *position*)**

Sets a column label or row label value.

If *orientation* is wxVERTICAL, the row label alignment is set. If *orientation* is wxHORIZONTAL, the column label alignment is set.

*position* is the label row or column position (starting from zero).

### **grid-set-row-height**

**void (grid-set-row-height long *grid-id* long *row* long *height*)** Sets the given row height in pixels.

### **grid-update-dimensions**

**void (grid-update-dimensions long *grid-id*)**

Recalculates dimensions so drawing is accurate. You may wish to call this if you alter a grid dimension, such as column width.

## **7.25. Groupbox**

A group box is a box drawn around one or more controls. Available under Windows only.

## group-box-create

**long** (**group-box-create** **long** *panel-id* **string** *label*  
**long** *x* **long** *y* **long** *width* **long** *height* **optional string** *style* **optional string** *name*)

Creates a group box and returns its id.

*name* gives the group box a name that can be retrieved with *window-get-name* (page 176).

## 7.26. Html

A subwindow used for displaying HTML files, using a class library written by Andrew Davison. At present only local HTML files should be loaded, and links in HTML files should again be local files, with non-URL specifications. Later releases will eventually allow Web browsing functionality, but to simplify wxCLIPS installation, this functionality has been omitted.

There are some bugs in scrolling and presentation, but for simple needs, it may prove handy to be able to show text and graphics (GIF files).

**Note:** this functionality is implemented in Windows only for the time being, and even then, not in all Windows releases of wxCLIPS and Hardy.

The following callbacks are valid for the html class.

**OnSize** The function is called with the window identifier, width and height.

**OnOpenURL** The function is called just before a URL is about to be opened, with the window identifier, and a URL. Return 1 to allow default processing, 0 to veto further processing. You can use this to program special URLs as buttons, if you test the URL and return 0 if you will process it yourself.

**OnSetStatusText** This is called with the window identifier and text, whenever it is appropriate to notify the user of the URL the mouse is over.

See also *window-add-callback* (page 175).

## html-back

**void** (**html-back** **long** *id*)

Loads and displays the previously-displayed URL or file.

## html-cancel

**void** (**html-cancel** **long** *id*)

Sets a flag to cancel the current operation.

## html-clear-cache

**void** (**html-clear-cache** **long** *id*)



Clears the internal cache.

### **html-create**

**long** (**html-create** **long** *parent-id* **optional long** *x* **optional long** *y* **optional long** *width* **optional long** *height* **optional string** *style=0* **optional string** *name="html"*)

Creates an HTML window.

*parent-id* must be a valid frame ID.

*name* gives the canvas a name that can be retrieved with *window-get-name* (page 176).

### **html-get-current-url**

**string** (**html-get-current-url** **long** *id*) Returns the current URL.

### **html-on-size**

**void** (**html-on-size** **long** *id* **long** *width* **long** *height*)

Invokes the HTML panel's OnSize member. This may need to be called if you override OnSize.

### **html-open-file**

**long** (**html-open-file** **long** *id* **string** *file*) Opens and displays a file.

### **html-resize**

**void** (**html-resize** **long** *id*)

Resizes and displays the current file.

### **html-save-file**

**long** (**html-save-file** **long** *id* **string** *file*) Saves the currently displayed file.

### **html-open-url**

**long** (**html-open-url** **long** *id* **string** *url*)

### **Opens a URL (not yet functioning).7.27. Icon**

An icon is a small bitmap which can be used to decorate a minimized frame. There are platform-specific ways of creating an icon.

## icon-create

**long (icon-create string *fileOrResource*)**

Creates an icon. Under X, the argument must be the filename of a valid XBM (X bitmap) file. Under Windows, the argument must be the name of an icon resource compiled into the current executable.

Use *frame-set-icon* (page 125) to set the icon of a frame.

## icon-delete

**long (icon-delete long *icon-id*)**

Deletes the given icon.

## icon-get-height

**long (icon-get-height long *icon-id*)**

Gets the height of the icon.

## icon-get-width

**long (icon-get-width long *icon-id*)**

Gets the width of the icon.

## icon-load-from-file

**long (icon-load-from-file string *file*, string *bitmap-type*)**

Loads an icon from a file. Under X, the argument must be the filename of a valid XBM (X bitmap) file. Under Windows, the argument must be the filename of a Windows icon file.

Under X, the permitted icon types in the *bitmap-type* are:

- **wxBITMAP\_TYPE\_BMP** Load a Windows bitmap file (if `USE_IMAGE_LOADING_IN_X` is enabled in `wx_setup.h`).
- **wxBITMAP\_TYPE\_GIF** Load a GIF bitmap file (if `USE_IMAGE_LOADING_IN_X` is enabled in `wx_setup.h`).
- **wxBITMAP\_TYPE\_XBM** Load an X bitmap file.
- **wxBITMAP\_TYPE\_XPM** Load an XPM (colour pixmap) file. Only available if `USE_XPM_IN_X` is enabled in `wx_setup.h`.

Under Windows, the permitted types are:

- **wxBITMAP\_TYPE\_ICO** Load a cursor from a `.ico` icon file (only if `USE_RESOURCE_LOADING_IN_MSW` is enabled in `wx_setup.h`).

- **wxBITMAP\_TYPE\_ICO\_RESOURCE** Load a Windows resource (as specified in the .rc file).

## 7.28. Instance table

wxCLIPS provides some functions for mapping between the integer identifiers used to represent objects in wxCLIPS functions, and COOL instance names. When creating object-oriented wrappers around wxCLIPS function groups, you can add an instance name entry in the **init** handler, and delete it in the **delete** handler. For each event, you can register a callback which retrieves the instance name from the identifier passed to the callback, and sends an appropriate message to that instance.

See also *wxclips-object-exists* (page 189).

### instance-table-add-entry

**long (instance-table-add-entry long *id* instance *instance-name*)**

Adds an entry to the instance table, indexing on the integer *id*.

### instance-table-delete-entry

**long (instance-table-delete-entry long *id*)**

Deletes an entry from the instance table.

### instance-table-get-instance

**instance-name (instance-table-get-instance long *id*)**

Retrieves an instance name for the integer *id*.

## 7.29. Key event

A key event identifier is passed to a window's OnChar or OnCharHook callback. The key code, position and state of shift/control/alt can be examined by calling the following functions.

### key-event-alt-down

**long (key-event-alt-down long *event-id*)**

Returns 1 if alt was pressed.

### key-event-control-down

**long (key-event-control-down long *event-id*)**

Returns 1 if control was pressed.

**key-event-get-key-code****string** (**key-event-get-key-code** **long** *event-id*)

Returns a string corresponding to the internal wxWindows key code, such as "WXK\_BACK", "WXK\_F1" or "WXK\_RETURN".

**key-event-position-x****double** (**key-event-position-x** **long** *event-id*)

Gets the x position of the mouse pointer at the moment the key was pressed.

**key-event-position-y****double** (**key-event-position-y** **long** *event-id*)

Gets the y position of the mouse pointer at the moment the key was pressed.

**key-event-shift-down****long** (**key-event-shift-down** **long** *event-id*)

Returns 1 if shift was pressed.

**7.30. Listbox**

A listbox displays a choice of strings. It must be the child of a *panel* (page 154). In a single-selection listbox, only one choice may be highlighted. In a multiple-selection listbox, several may be highlighted.

**list-box-create****long** (**list-box-create** **long** *panel-id* **string** *callback* **string** *label*  
**long** *multiple*, **optional long** *x* **optional long** *y*  
**optional long** *width* **optional long** *height* **optional string** *style* **optional string** *name*)

Creates a list box item on the given panel. The callback may be the empty string (""), to denote no callback, or a word or string for the function name. The function will be called when an item in the list box is selected or deselected, with the list box ID as argument. The value of *multiple* should be 1 if multiple selections are required, or 0 if only a single selection is required.

*style* is a *bit list* of some of the following:

wxNEEDED\_SB Create scrollbars if needed.

wxALWAYS\_SB Create scrollbars immediately.

wxHSCROLL Create horizontal scrollbar if contents are too wide (Windows only).

*name* gives the group box a name that can be retrieved with *window-get-name* (page 176).

If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

### **list-box-append**

**long** (**list-box-append** **long** *list-box-id* **string** *item*  
**optional string** *client-data*)

Append a string to the list box, with an optional client data string.

### **list-box-find-string**

**long** (**list-box-find-string** **long** *list-box-id* **string** *item*)

Find the string in the list box and return the integer position if found, -1 if not.

### **list-box-clear**

**long** (**list-box-clear** **long** *list-box-id*)

Clear all strings from the list box.

### **list-box-get-selection**

**long** (**list-box-get-selection** **long** *list-box-id*)

Get the position of the selection (for single-selection list boxes only).

### **list-box-get-string-selection**

**string** (**list-box-get-string-selection** **long** *list-box-id*)

Get the selected string (for single-selection list boxes only).

### **list-box-is-selected**

**long** (**list-box-is-selected** **long** *list-box-id* **long** *item*)

Returns 1 if *item* is selected, 0 otherwise.

### **list-box-set-selection**

**long** (**list-box-set-selection** **long** *list-box-id* **long** *item-pos* **long** *flag=1*)

Set a selection by item position.

If *flag* is 1, the item is selected, otherwise it is deselected.

### **list-box-set-string-selection**

**long** (**list-box-set-string-selection** **long** *list-box-id* **string** *item*)

Set a selection by string.

### **list-box-number**

**long** (**list-box-number** **long** *list-box-id*)

Return the number of items in the list box.

### **list-box-delete**

**long** (**list-box-delete** **long** *list-box-id* **long** *item-pos*)

Delete an item in the list box.

### **list-box-get-string**

**string** (**list-box-get-string** **long** *list-box-id* **long** *item-pos*)

Return the string at the given position.

### **list-box-get-first-selection**

**long** (**list-box-get-first-selection** **long** *list-box-id*)

Get the first selection position in a multi-selection list box (-1 for no more selections).

### **list-box-get-next-selection**

**long** (**list-box-get-next-selection**)

Get the next selection position in a multi-selection list box (-1 for no more selections).

## **7.31. Memory device context**

A memory device context is used for drawing into, or copying from, a bitmap. See also the *Bitmap* (page 93) object.

### **memory-dc-create**

**long** (**memory-dc-create**)

Create a memory device context and returns its ID.

### **memory-dc-select-object**

**long (memory-dc-select-object long *id* long *bitmap-id*)**

Makes this device context the drawing surface for the given bitmap (see *Bitmap* (page 93)). Deleting the memory device context disassociates the bitmap, freeing it to be used with another memory device context. To draw a bitmap on a device context that supports bitmap drawing (i.e. not a Metafile or PostScript device context), using code like the following:

```
;;; Utility function for drawing a bitmap
(deffunction draw-bitmap (?dc ?bitmap ?x ?y)
  (bind ?mem-dc (memory-dc-create))
  (memory-dc-select-object ?mem-dc ?bitmap)
  ; Blit the memory device context onto the destination device context
  (dc-blit ?dc ?x ?y (bitmap-get-width ?bitmap) (bitmap-get-height
?bitmap)
    ?mem-dc 0.0 0.0)
  (memory-dc-delete ?mem-dc)
)
```

If *bitmap-id* is zero, the existing bitmap (if any) will be selected out of the device context. This might be necessary if you wish to delete the bitmap before deleting the device context (for example, for reusing the same device context for different bitmaps).

## **7.32. Menu**

The menu is used only as a component of a *menu bar* (page 145). Create menus, append menu items (strings, separators or further menus), and finally append the menu to the menu bar.

A menu or menu bar string may contain an ampersand, which is taken to mean 'underline the next character and use it as the hotkey'. This gives the user the opportunity to use keystrokes to access menus and items.

### **menu-create**

**long (menu-create optional string *label* optional string *callback*)**

Create a menu and returns the menu's ID.

*label* is unused at present.

*callback* should be present if creating a popup menu (i.e. not a menubar menu). It will be called with the menu's id when the user selects an item. From within the callback, use *panel-item-get-command-event* (page 155) to retrieve the command event and from that, the menu item selection.

### **menu-append**

**long (menu-append long *menu-id* long *item-id*  
string *item-string* optional long *submenu-id* optional string *help-string* optional long**

*checkable*)

Append a string or submenu to the menu, passing the integer ID by which the menu item will be referenced, a string to be displayed, an optional id for a pullright menu, and an optional flag for specifying whether this menu item can be checked.

A help string can be supplied, in which case the string will be shown on the first field of the status line (if any) in the frame containing the menu bar, when the mouse pointer moves over the menu item.

### **menu-append-separator**

**long (menu-append-separator long *menu-id*)**

Append a menu separator.

### **menu-break**

**long (menu-break long *menu-id*)**

Inserts a column break into the menu.

### **menu-check**

**long (menu-check long *menu-id* long *item-id* long *check*)**

Check (*check* = 1 or uncheck *check* = 0 the given menu item. MS Windows only.

### **menu-enable**

**long (menu-enable long *menu-id* long *item-id* long *enable*)**

Enable (*enable* = 1 or disable *enable* = 0 the given menu item.

## **7.33. Menu bar**

A menu bar is a standard user interface element which places the main commands of an application along the top of a *frame* (page 123).

The menu bar must be assigned to a frame using *frame-set-menu-bar* (page 125). Once this is done, the menu bar must not be deleted by the application: it will be deleted when the frame is deleted.

A menu or menu bar string may contain an ampersand, which is taken to mean 'underline the next character and use it as the hotkey'. This gives the user the opportunity to use keystrokes to access menus and items.

See also *Menu* (page 144).

### **menu-bar-create**



**long (menu-bar-create)**

Create a menu bar and return its ID.

**menu-bar-create-from-resource**

**long (menu-bar-create-from-resource string *resource-name*)**

Create a menu bar and return its ID, given a resource name.

The resource file containing this resource must first have been loaded with *load-resource-file* (page 185).

**menu-bar-append**

**long (menu-bar-append long *menu-bar-id* long *menu-id* string *title*)**

Append a menu to a menu bar.

**menu-bar-check**

**long (menu-bar-check long *menu-bar-id* long *item-id* long *check*)**

Check (*check* = 1) or uncheck (*check* = 0) the given menu item (MS Windows only).

**menu-bar-checked**

**long (menu-bar-checked long *menu-bar-id* long *item-id*)**

Returns 1 if the menu item is checked, 0 otherwise.

**menu-bar-enable**

**long (menu-bar-enable long *menu-bar-id* long *item-id* long *enable*)**

Enable (*enable* = 1) or disable (*enable* = 0) the given menu item.

**7.34. Message**

A message is a simple piece of text on a *panel* (page 154).

**message-create**

**long (message-create long *panel-id* string *label*  
optional long *x* optional long *y*  
optional string *style* optional string *name*)**

Creates a message item on the given panel (a simple, non-selectable, non-editable string). If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

*style* is reserved for future use.

*name* gives the message a name that can be retrieved with *window-get-name* (page 176).

### **message-create-from-bitmap**

**long (message-create-from-bitmap long *panel-id* long *bitmap-id*  
optional long *x* optional long *y*  
optional long *width* optional long *height* optional string *style* optional string *name*)**

Creates a bitmap message given a valid bitmap identifier.

*name* gives the message a name that can be retrieved with *window-get-name* (page 176).

## **7.35. Metafile**

A metafile is the Windows vector format. Currently, the only way of creating a Windows metafile is to close a metafile device context, and the only valid operations are to delete the metafile and to place it on the clipboard.

These functions are only available under Windows.

### **7.35.1. Example**

Below is a example of metafile, metafile device context and clipboard use. Note the way the metafile dimensions are passed to the clipboard, making use of the device context's ability to keep track of the maximum extent of drawing commands.

```
(bind ?dc (metafile-dc-create))
(if (eq (dc-ok ?dc) 1) then
  (
    ; Do some drawing
    (bind ?mf (metafile-dc-close ?dc))
    (if (neq ?mf 0) then
      ; Pass metafile to the clipboard
      (metafile-set-clipboard ?mf (dc-get-max-x ?dc) (dc-get-max-y
?dc))
      (metafile-delete ?mf)
    )
  )
)
(dc-delete ?dc)
```

### **metafile-delete**

**long (metafile-delete long *id*)**

Deletes the metafile.

**metafile-set-clipboard****long (metafile-set-clipboard long *id* int *width* int *height*)**

Places the metafile on the clipboard, returning 1 for success and 0 for failure.

The metafile should be deleted immediately after this operation.

**7.36. Metafile device context**

A metafile device context is used for creating a metafile. The programmer should create the metafile device context, close it to return a metafile, delete the device context, use the metafile (the only valid thing to do with it currently is to place it on the clipboard, and then delete the metafile).

These functions are only available under Windows.

See also *Metafile* (page 147).

**metafile-dc-create****long (metafile-dc-create optional string *filename*)**

Creates a metafile device context and returns its ID.

*filename* is the file to be used if creating a disk-based metafile. Usually this will be zero or absent, and an in-memory metafile will be created.

**metafile-dc-close****long (metafile-dc-close long *id*)**

Closes the metafile device context and returns a metafile. The device context should no longer be used after this call is made, and it should be deleted.

See *Metafile* (page 147).

**7.37. Mouse event**

A mouse event identifier is passed to the canvas *OnEvent* (page 175) callback. The state of the mouse buttons (and some keys) can be examined by calling the following functions.

**mouse-event-button****long (mouse-event-button long *event-id* long *button*)**

Returns 1 if the given button is changing state. *button* may be 1, 2 or 3 (left, middle and right buttons respectively).

**mouse-event-button-down****long (mouse-event-button-down long *event-id*)**

Returns 1 if the event is a mouse button down event.

**mouse-event-control-down****long (mouse-event-control-down long *event-id*)**

Returns 1 if the control key is down.

**mouse-event-dragging****long (mouse-event-dragging long *event-id*)**

Returns 1 if the event is a dragging event (holding a mouse button down and moving).

**mouse-event-left-down****long (mouse-event-left-down long *event-id*)**

Returns 1 if the left mouse button is down.

**mouse-event-left-up****long (mouse-event-left-up long *event-id*)**

Returns 1 if the left mouse button is up.

**mouse-event-is-button****long (mouse-event-is-button long *event-id*)**

Returns 1 if the event is a button press or release.

**mouse-event-middle-down****long (mouse-event-middle-down long *event-id*)**

Returns 1 if the middle mouse button is down.

**mouse-event-middle-up****long (mouse-event-middle-up long *event-id*)**

Returns 1 if the middle mouse button is up.

**mouse-event-position-x****double** (**mouse-event-position-x** **long** *event-id*)

Returns the mouse x-position.

**mouse-event-position-y****double** (**mouse-event-position-y** **long** *event-id*)

Returns the mouse y-position.

**mouse-event-right-down****long** (**mouse-event-right-down** **long** *event-id*)

Returns 1 if the right mouse button is down.

**mouse-event-right-up****long** (**mouse-event-right-up** **long** *event-id*)

Returns 1 if the right mouse button is up.

**mouse-event-shift-down****long** (**mouse-event-shift-down** **long** *event-id*)

Returns 1 if the shift key is down.

**7.38. Multi-line text**

A multi-line text item is able to show several lines of text, unlike the single line *text* (page 166) item. It must be the child of a *panel* (page 154).

Under Windows, there is an extended range of functions. Some take character positions - a single integer which can identify a character position - and others take line and character numbers. If you want to use a function that takes one form, but you only have the other, you can convert between them using a function such as *multi-text-xy-to-position* or *multi-text-position-to-line*. Note that line and character numbers start from zero.

**multi-text-create****long** (**multi-text-create** **long** *panel-id* **string** *callback* **string** *label*  
**optional string** *value* **optional long** *x* **optional long** *y*  
**optional long** *width* **optional long** *height* **optional string** *style* **optional string** *name*)

Creates a multi-line text item on the given panel. The callback may be the empty string ("") to denote no callback, or a word or string for the function name. The function will be called when return is pressed in the text item, with the text item ID as argument. The default value is optional.

If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

The *style* parameter can be a *bit list* of the following:

**wxHSCROLL**    A horizontal scrollbar will be displayed. If **wxHSCROLL** is omitted, only a vertical scrollbar is displayed, and lines will be wrapped. This parameter is ignored under XView.  
**wxREADONLY**    The text is read-only (not XView).

*name* gives the multitext a name that can be retrieved with *window-get-name* (page 176).

### **multi-text-copy**

**long (multi-text-copy long window-id)**

Copies the selected text to the clipboard. Windows only.

### **multi-text-cut**

**long (multi-text-cut long window-id)**

Copies the selected text to the clipboard, then removes the selection. Windows only.

### **multi-text-get-insertion-point**

**long (multi-text-get-insertion-point long window-id)**

Returns the insertion point. Windows only.

### **multi-text-get-last-position**

**long (multi-text-get-last-position long window-id)**

Returns the final position in the text window. Windows only.

### **multi-text-get-line-length**

**long (multi-text-get-line-length long window-id long line-no)**

Returns the length of the text at line *line-no*. Windows only.

### **multi-text-get-line-length**

**long (multi-text-get-line-text long window-id long line-no)**

Returns the text at *line-no*.

### **multi-text-get-number-of-lines**

**long (multi-text-get-number-of-lines long window-id)**

Returns the number of lines in the text window. Windows only.

### **multi-text-get-value**

**string (multi-text-get-value long multi-text-item)**

Get the multi-text item's string value.

### **multi-text-set-value**

**long (multi-text-set-value long multi-text-item string value)**

Set the multi-text item's string value.

### **multi-text-paste**

**long (multi-text-paste long window-id)**

Pastes the text (if any) from the clipboard to the text window. Windows only.

### **multi-text-position-to-char**

**long (multi-text-position-to-char long window-id long pos)**

Returns the character position (starting from zero) for the given index position. Windows only.

### **multi-text-position-to-line**

**long (multi-text-position-to-line long window-id long pos)**

Returns the line number (starting from zero) for the given index position. Windows only.

### **multi-text-remove**

**long (multi-text-remove long window-id long start-pos long end-pos)**

Removes the text between the given span selection. Windows only.

**multi-text-replace**

**long** (**multi-text-replace** **long** *window-id* **long** *start-pos* **long** *end-pos* **string** *text*)

Replaces the text between the given span selection with the given text.

**multi-text-set-insertion-point**

**long** (**multi-text-set-insertion-point** **long** *window-id* **long** *pos*)

Sets the insertion point to the given index position. Windows only.

**multi-text-set-selection**

**long** (**multi-text-set-selection** **long** *window-id* **long** *start-pos* **long** *end-pos*)

Sets the selection to the given span of text. Windows only.

**multi-text-show-position**

**long** (**multi-text-show-position** **long** *window-id* **long** *pos*)

Shows the text at the given index position. Windows only.

**multi-text-write**

**long** (**multi-text-write** **long** *window-id* **string** *text*)

Writes the given string into the multitext, at the current cursor point. Windows only.

**multi-text-xy-to-position**

**long** (**multi-text-xy-to-position** **long** *window-id* **long** *char-position* **long** *line*)

Converts the character and line number (each starting from zero) to a position.

**7.39. Object**

An object is a general term for any wxCLIPS entity, such as window, brush, pen, listbox, etc.

**object-delete**

**long** (**object-delete** **long** *id*)

Deletes an object.

**object-get-type**



**char \* (object-get-type long id)**

Returns the C++ class name for the object.

## 7.40. Panel

A panel is a subwindow for placing panel items, such as *buttons* (page 95) and *text items* (page 166). Its parent must be a *frame* (page 123). A panel inherits most properties from canvas, except for scrollbar functionality.

Note that a *dialog box* (page 121) may be used in a similar way to a panel.

The following callbacks are valid for the panel class:

**OnCommand** Called with a panel identifier, an item identifier and a command event identifier when a command event is received by a panel item that does not have an associated callback. If you have created a panel or dialog box from a resource, you will need to intercept OnCommand.

**OnDefaultAction** Called with a panel identifier and an item identifier, when a double click has been received from a listbox.

**OnEvent** Called with a panel identifier and a *mouse event* (page 148) identifier. This can only be guaranteed only when the panel is in user edit mode (to be implemented).

**OnPaint** Called with a panel identifier when the panel receives a repaint event from the window manager.

**OnSize** The function is called with the window identifier, width and height.

See also *window-add-callback* (page 175).

## panel-create

**long (panel-create long parent-id optional long x optional long y  
optional long width optional long height optional string style optional string name)**

Creates a panel. *parent-id* must be a valid frame ID.

The *style* parameter may be a combination of the following, using the bitwise 'or' operator.

**wxABSOLUTE\_POSITIONING** A hint to the windowing system not to try native Windowing system layout (Motif only). This is the recommended style for all Motif panels and dialog boxes.

**wxBORDER** Draws a thin border around the panel.

**wxVSCROLL** Gives the dialog box a vertical scrollbar (XView only).

*name* gives the panel a name that can be retrieved with *window-get-name* (page 176).

## panel-create-from-resource

**long (panel-create-from-resource long parent-id string resource-name)**

Creates a panel from the given wxWindows resource. The resource file containing this resource must first have been loaded with *load-resource-file* (page 185).

Panel items on a panel or dialog box that has been created from a resource, do not have conventional callbacks. Therefore you need to intercept the `OnCommand` event for the panel or dialog box and test the name and event of the item passed to this callback.

### **panel-set-button-font**

**long** (**panel-set-button-font** long *panel-id* long *font-id*)

Sets the font used for panel or dialog box item buttons (or contents). See also *panel-set-label-font* (page 155).

### **panel-set-label-font**

**long** (**panel-set-label-font** long *panel-id* long *font-id*)

Sets the font used for panel or dialog box item labels. See also *panel-set-button-font* (page 155).

### **panel-set-label-position**

**long** (**panel-set-label-position** long *panel-id* string *position*)

Change the current label orientation for panel items: *position* may be `wxVERTICAL` or `wxHORIZONTAL`.

### **panel-new-line**

**long** (**panel-new-line** long *panel-id*)

Insert a new line, that is, make subsequent panel items appear at the start of the next line.

## **7.41. Panel item**

A panel item is a control (or widget) that can be placed on a *panel* (page 154) to accept user input, and display information.

The following functions apply to panel items, which include *button* (page 95), *checkbox* (page 100), *choice* (page 100), *message* (page 146), *text* (page 166), *multi-line text* (page 150), *slider* (page 165).

### **panel-item-get-command-event**

**long** (**panel-item-get-command-event**)

Returns the identifier of the command event for the current panel item or menu callback, or zero if not called within a callback.

### **panel-item-get-label**

**string** (**panel-item-get-label** **long** *panel-id*)

Get the item's label.

### **panel-item-set-default**

**long** (**panel-item-set-default** **long** *panel-id*)

Make this item the default.

### **panel-item-set-label**

**long** (**panel-item-set-label** **long** *panel-id* **string** *label*)

Set the item's label.

## **7.42. Pen**

A pen is used to control the colour and style of subsequent drawing operations on a *device context* (page 115).

### **pen-create**

**long** (**pen-create** **string** *colour* **long** *width* **word** *style*)

**long** (**pen-create** **long** *colour-value* **long** *width* **word** *style*)

Creates a pen for use in a device context. A pen is used for the outlines of graphic shapes. A brush must be set to fill the shapes.

*colour* is a wxWindows colour string such as "BLACK", "CYAN".

*colour-value* is a value returned from *colour-create* (page 103).

*width* specifies the width of the pen.

*style* may be one of wxSOLID, wxDOT, wxLONG\_DASH, wxSHORT\_DASH, wxTRANSPARENT.

### **pen-delete**

**long** (**pen-delete** **long** *pen-id*)

Deletes the given pen.

## **7.43. PostScript device context**

A PostScript device context is used for drawing into a postscript file.

## postscript-dc-create

**long** (**postscript-dc-create** **optional string** *file*  
**optional long** *interactive* **optional long** *window-id*)

Creates a postscript device context and returns its ID.

*file* is the file to be used for printing to. *interactive* may be 1 to popup up a printer dialog, or 0 otherwise. *window-id* is a parent window for the printer dialog.

## 7.44. Printer device context

A Printer device context is used for drawing onto a Windows printer.

## printer-dc-create

**long** (**printer-dc-create** **optional string** *driver* **optional string** *device*  
**optional string** *filename* **optional long** *interactive*)

Creates a printer device context and returns its ID.

*file* is the file to be used for printing to. *interactive* may be 1 to popup up a printer dialog, or 0 otherwise.

## 7.45. Radiobox

A radiobox item is a matrix of strings with associated radio buttons. The buttons are mutually exclusive, so pressing one will deselect the current selection.

## radio-box-create

**long** (**radio-box-create** **long** *panel-id* **string** *callback* **string** *label*  
**long** *x* **long** *y* **long** *width* **long** *height*  
**multivalue** *strings* **long** *major-dimension* **optional string** *style* **optional string** *name*)

Creates a radiobox item on the given panel. The callback may be the empty string (""), to denote no callback, or a word or string for the function name. The function will be called when an item in the radiobox is selected, with the radiobox ID as argument. If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

*strings* should be a multifield of strings.

*major-dimension* specifies the number of rows (if style is wxVERTICAL) or columns (if style is wxHORIZONTAL) for a two-dimensional radiobox.

*style* specifies a *bit list* of styles.

wxVERTICAL    Lays the radiobox out in columns.

wxHORIZONTAL    Lays the radiobox out in rows.

*name* gives the radiobox a name that can be retrieved with *window-get-name* (page 176).

**radio-box-get-selection****long (radio-box-get-selection long *radio-box-id*)**

Get the ID of the button currently selected.

**radio-box-set-selection****long (radio-box-set-selection long *radio-box-id* long *item*)**

Sets the given button to be the current selection.

**7.46. Recordset**

See also *Database classes overview* (page 205)

Each recordset represents an ODBC database query. You can make multiple queries at a time by using multiple recordsets with a database or you can make your queries in sequential order using the same recordset.

**recordset-create****long (recordset-create long *db* optional string *type* = "wxOPEN\_TYPE\_DYNASET" optional string *options* = "wxOPTION\_DEFAULT")**

Constructs a recordset object and returns its id. *db* is a pointer to the database instance you wish to use the recordset with. Currently there are two possible values of *type*:

- "wxOPEN\_TYPE\_DYNASET": Loads only one record at a time into memory. The other data of the result set will be loaded dynamically when moving the cursor. This is the default type.
- "wxOPEN\_TYPE\_SNAPSHOT": Loads all records of a result set at once. This will need much more memory, but will result in faster access to the ODBC data.

The *options* parameter is not used yet.

The function appends the recordset object to the parent database's list of recordset objects, for later destruction when the database is destroyed.

**recordset-delete****long (recordset-delete long *id*)**

Deletes the recordset. All data except that stored in user-defined variables will be lost. It also unlinks the recordset object from the parent database's list of recordset objects.

**recordset-execute-sql****long (recordset-execute-sql long *id* string *sql*)**

Directly executes a SQL statement. The data will be presented as a normal result set. Note that the recordset must have been created as a snapshot, not dynaset. Dynasets will be implemented in the near future.

Examples of common SQL statements are given in *A selection of SQL commands* (page 209).

### **recordset-get-char-data**

**string (recordset-get-char-data long *id* string-or-long *col*)**

Returns the character (string) data for the current record at the specified column. The column can be a name or an integer position (starting from zero).

### **recordset-get-col-name**

**string (recordset-get-col-name long *id* long *col*)**

Gets the name of the column at position *col*. Returns the empty string if *col* does not exist.

### **recordset-get-col-type**

**string (recordset-get-col-type long *id* string-or-long *col*)**

Gets the name of the column at position *col* or name *col*. Returns "SQL\_TYPE\_NULL" if *col* does not exist.

See *ODBC SQL data types* (page 208) for the possible return values from this function.

### **recordset-get-columns**

**long (recordset-get-columns long *id* optional string *table* = "")**

Returns the columns of the table with the specified name. If no name is given, the internal class member *table* will be used. If both names are NULL nothing will happen. The data will be presented as a normal result set, organized as follows:

0 (VARCHAR) TABLE\_QUALIFIER

1 (VARCHAR) TABLE\_OWNER

2 (VARCHAR) TABLE\_NAME

3 (VARCHAR) COLUMN\_NAME

4 (SMALLINT) DATA\_TYPE

5 (VARCHAR) TYPE\_NAME

6 (INTEGER) PRECISION

7 (INTEGER)    LENGTH  
8 (SMALLINT)   SCALE  
9 (SMALLINT)   RADIX  
10 (SMALLINT)   NULLABLE  
11 (VARCHAR)   REMARKS

### **recordset-get-database**

**long (recordset-get-database long *id*)**

Returns the identifier of the parent database.

### **recordset-get-data-sources**

**long (recordset-get-data-sources long *id*)**

Gets the currently-defined data sources via the ODBC manager. The data will be presented as a normal result set. See the documentation for the ODBC function `SQLDataSources` for how the data is organized. The name of the source is at column 0.

### **recordset-get-error-code**

**string (recordset-get-error-code long *id*)**

Returns the error code of the last ODBC action. This will be a string containing one of:

SQL\_ERROR    General error.  
SQL\_INVALID\_HANDLE    An invalid handle was passed to an ODBC function.  
SQL\_NEED\_DATA    ODBC expected some data.  
SQL\_NO\_DATA\_FOUND    No data was found by this ODBC call.  
SQL\_SUCCESS    The call was successful.  
SQL\_SUCCESS\_WITH\_INFO    The call was successful, but further information can be obtained from the ODBC manager.

### **recordset-get-filter**

**string (recordset-get-filter long *id*)**

Returns the current filter.

### **recordset-get-float-data**

**double (recordset-get-float-data long *id* string-or-long *col*)**

Returns the floating-point data for the current record at the specified column. The column can be a name or an integer position (starting from zero).

### **recordset-get-foreign-keys**

**long (recordset-get-foreign-keys long *id* optional string *f*table = "" optional string *k*table = "")**

Returns a list of foreign keys in the specified table (columns in the specified table that refer to primary keys in other tables), or a list of foreign keys in other tables that refer to the primary key in the specified table.

If *p*table contains a table name, this function returns a result set containing the primary key of the specified table.

If *f*table contains a table name, this functions returns a result set of containing all of the foreign keys in the specified table and the primary keys (in other tables) to which they refer.

If both *p*table and *f*table contain table names, this function returns the foreign keys in the table specified in *f*table that refer to the primary key of the table specified in *p*table. This should be one key at most.

GetForeignKeys returns results as a standard result set. If the foreign keys associated with a primary key are requested, the result set is ordered by FKTABLE\_QUALIFIER, FKTABLE\_OWNER, FKTABLE\_NAME, and KEY\_SEQ. If the primary keys associated with a foreign key are requested, the result set is ordered by PKTABLE\_QUALIFIER, PKTABLE\_OWNER, PKTABLE\_NAME, and KEY\_SEQ. The following table lists the columns in the result set.

0 (VARCHAR)	PKTABLE_QUALIFIER
1 (VARCHAR)	PKTABLE_OWNER
2 (VARCHAR)	PKTABLE_NAME
3 (VARCHAR)	PKCOLUMN_NAME
4 (VARCHAR)	FKTABLE_QUALIFIER
5 (VARCHAR)	FKTABLE_OWNER
6 (VARCHAR)	FKTABLE_NAME
7 (VARCHAR)	FKCOLUMN_NAME
8 (SMALLINT)	KEY_SEQ
9 (SMALLINT)	UPDATE_RULE
10 (SMALLINT)	DELETE_RULE
11 (VARCHAR)	FK_NAME
12 (VARCHAR)	PK_NAME

### **recordset-get-int-data**

**long (recordset-get-int-data long *id* string-or-long *col*)**

Returns the integer data for the current record at the specified column. The column can be a name or an integer position (starting from zero).

### **recordset-get-number-cols**



**long (recordset-get-number-cols long *id*)**

Returns the number of columns in the result set.

### **recordset-get-number-fields**

**long (recordset-get-number-fields long *id*)**

Not implemented.

### **recordset-get-number-params**

**long (recordset-get-number-params long *id*)**

Not implemented.

### **recordset-get-number-records**

**long (recordset-get-number-records long *id*)**

Returns the number of records in the result set.

### **recordset-get-primary-keys**

**long (recordset-get-primary-keys long *id* optional string *table* = "")**

Returns the column names that comprise the primary key of the table with the specified name. If no name is given the class member *tablename* will be used. If both names are NULL nothing will happen. The data will be presented as a normal result set, organized as follows:

0 (VARCHAR)	TABLE_QUALIFIER
1 (VARCHAR)	TABLE_OWNER
2 (VARCHAR)	TABLE_NAME
3 (VARCHAR)	COLUMN_NAME
4 (SMALLINT)	KEY_SEQ
5 (VARCHAR)	PK_NAME

### **recordset-get-result-set**

**long (recordset-get-result-set long *id*)**

Copies the data presented by ODBC into the recordset. Depending on the recordset type all or only one record(s) will be copied. Usually this function will be called automatically after each successful database operation.

### **recordset-get-table-name**

**string (recordset-get-table-name long *id*)**

Returns the name of the current table.

### **recordset-get-tables**

**long (recordset-get-tables long *id*)**

Gets the tables of a database. The data will be presented as a normal result set, organized as follows:

0 (VARCHAR)	TABLE_QUALIFIER
1 (VARCHAR)	TABLE_OWNER
2 (VARCHAR)	TABLE_NAME
3 (VARCHAR)	TABLE_TYPE (TABLE, VIEW, SYSTEM TABLE, GLOBAL TEMPORARY, LOCAL TEMPORARY, ALIAS, SYNONYM, or database-specific type)
4 (VARCHAR)	REMARKS

### **recordset-goto**

**long (recordset-goto long *id* long *n*)**

Moves the cursor to the record with the number *n*, where the first record has the number 0.

### **recordset-is-bof**

**long (recordset-is-bof long *id*)**

Returns 1 if the user tried to move the cursor before the first record in the set.

### **recordset-is-field-dirty**

**long (recordset-is-field-dirty long *id* string-or-long *field*)**

Returns 1 if the given field has been changed but not saved yet.

### **recordset-is-field-null**

**long (recordset-is-field-null long *id* string-or-long *field*)**

Returns 1 if the given field has no data.

### **recordset-is-col-nullable**

**long (recordset-is-col-nullable long *id* string-or-long *field*)**

Returns 1 if the given column may contain no data.

**recordset-is-eof****long (recordset-is-eof long *id*)**

Returns 1 if the user tried to move the cursor behind the last record in the set.

**recordset-is-open****long (recordset-is-open long *id*)**

Returns 1 if the parent database is open.

**recordset-move****long (recordset-move long *id* long *rows*)**

Moves the cursor a given number of rows. Negative values are allowed.

**recordset-move-first****long (recordset-move-first long *id*)**

Moves the cursor to the first record.

**recordset-move-last****long (recordset-move-last long *id*)**

Moves the cursor to the last record.

**recordset-move-next****long (recordset-move-next long *id*)**

Moves the cursor to the next record.

**recordset-move-prev****long (recordset-move-prev long *id*)**

Moves the cursor to the previous record.

**recordset-query****long (recordset-query long *id* string *columns* string *table* optional string *filter*)**

Start a query. An SQL string of the following type will automatically be generated and executed: "SELECT columns FROM table WHERE filter".

**recordset-set-table-name**

**long (recordset-set-table-name long id string table)**

Specify the name of the table you want to use.

## 7.47. Server

See also *Interprocess communication overview* (page 199)

A server object represents the server side of a DDE conversation.

To delete a server object, use `object-delete`.

### **server-create**

**long (server-create string service-name)**

Creates a server object, and returns an integer id if successful.

*service-name* is a string identifying this service to potential clients. Under UNIX, it should contain a valid port number.

The application should use *window-add-callback* (page 175) to register the window callback `OnAcceptConnection` or `OnAcceptConnectionEx`, which will be called when a client requests a connection.

`OnAcceptConnection` will be called with arguments:

1. server id (long)
2. the name of the topic in which the client is interested (string)
3. tentative connection id (long)

If this function returns zero, the connection is rejected and deleted, otherwise it is confirmed. See also *connection* (page 104).

`OnAcceptConnectionEx` will be called with arguments:

1. server id (long)
2. the name of the topic in which the client is interested (string)

This form assumes that the connection object will be created with `connection-create` from within the callback.

## 7.48. Slider

A slider is a panel item for denoting a range of values. It must be a child of a *panel* (page 154).

### **slider-create**

**long (slider-create long panel-id string callback string label  
long value long min-value long max-value  
long width optional long x optional long y optional string style optional string name)**

Creates a horizontal slider item on the given panel. The callback may be the empty string (""), to denote no callback, or a word or string for the function name. The function will be called when the slider value is changed, with the slider item ID as argument.

If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

*style* is a *bit list* of the following:

wxHORIZONTAL      The item will be created as a horizontal slider.  
wxVERTICAL      The item will be created as a vertical slider.

*name* gives the slider a name that can be retrieved with *window-get-name* (page 176).

### slider-set-value

**long (slider-set-value long slider-id long value)**

Set the value of the slider.

### slider-get-value

**long (slider-get-value long slider-id)**

Gets the value of the slider.

## 7.49. Text

A text item is used for displaying and editing a single line of text. It must be a child of a *panel* (page 154).

See also *multi-line text* (page 150).

### text-create

**long (text-create long panel-id string callback string label  
optional string value optional long x optional long y  
optional long width optional long height optional string style optional string name)**

Creates a single-line text item on the given panel. The callback may be the empty string (""), to denote no callback, or a word or string for the function name. The function will be called when return is pressed in the text item, with the text item ID as argument. The default value is optional.

If no position is given, the panel item is placed after the last item. The value -1 may be passed to denote a default, so that the position may be left unspecified and the size given.

*style* may be the empty string, or a *bit list* of:

wxTE\_PROCESS\_ENTER      The callback function will receive the event  
wxEVENT\_TYPE\_TEXT\_ENTER\_COMMAND. Note that this will break tab  
traversal for this panel item under Windows. Single-line text only.

**wxTE\_PASSWORD**     The text will be echoed as asterisks. Single-line text only.  
**wxTE\_READONLY**     The text will not be user-editable.  
**wxHSCROLL**     A horizontal scrollbar will be displayed. If **wxHSCROLL** is omitted, only a vertical scrollbar is displayed, and lines will be wrapped. This parameter is ignored under XView. Multi-line text only.

*name* gives the group box a name that can be retrieved with *window-get-name* (page 176).

## **text-set-value**

**long (text-set-value long text-id string value)**

Set the string value of a text item.

## **text-get-value**

**string (text-get-value long text-id)**

Get the string value of a text item.

## **7.50. Text window**

To display a lot of text, use this subwindow as the child of a *frame* (page 123). It is capable of loading and saving files of ASCII text, and under Open Look and Motif, the text can be edited directly.

To allow the user to edit text under Windows as well as the other platforms, either invoke an external editor or create a *multi-line text item* (page 150) on a panel.

Under Windows, there is an extended range of functions. Some take character positions - a single integer which can identify a character position - and others take line and character numbers. If you want to use a function that takes one form, but you only have the other, you can convert between them using a function such as *text-window-xy-to-position* or *text-window-position-to-line*. Note that line and character numbers start from zero.

The following callbacks are valid for the dialog box class:

**OnChar** (Not XView.) The function is called with the text window identifier, key code, and key event identifier. If the event is an ASCII keypress, the code will correspond to the ASCII code; otherwise, the programmer must refer to the constants defined in `common.h`, in the `wxWindows` library.

To invoke default processing, call *text-window-on-char*.

**OnSize** The function is called with the text window identifier, width and height.

See also *window-add-callback* (page 175).

## **text-window-clear**

**long (text-window-clear long window-id)**

Clears the contents of a text subwindow. Returns 1 if successful, 0 otherwise.

### **text-window-copy**

**long (text-window-copy long window-id)**

Copies the selected text to the clipboard.

### **text-window-cut**

**long (text-window-cut long window-id)**

Copies the selected text to the clipboard, then removes the selection.

### **text-window-create**

**long (text-window-create long parent-id optional long x optional long y  
optional long width optional long height optional string style optional string name)**

Creates a text subwindow. *parent-id* must be a valid frame ID.

*style* is a *bit list* of some of the following:

wxBORDER	Use this style to draw a thin border in Windows 3 (non-native implementation only).
wxNATIVE_IMPL	Use this style to allow editing under MS Windows, albeit with a 64K limitation.

*name* gives the text window a name that can be retrieved with *window-get-name* (page 176).

### **text-window-discard-edits**

**void (text-window-discard-edits long window-id)**

Discard any edits in the text window.

### **text-window-get-contents**

**string (text-window-get-contents long window-id)**

Returns the window contents (to a maximum of 1000 characters).

### **text-window-get-insertion-point**

**long (text-window-get-insertion-point long window-id)**

Returns the insertion point.

**text-window-get-last-position****long** (**text-window-get-last-position** **long** *window-id*)

Returns the final position in the text window.

**text-window-get-line-length****long** (**text-window-get-line-length** **long** *window-id* **long** *line-no*)

Returns the length of the text at line *line-no*.

**text-window-get-line-text****long** (**text-window-get-line-text** **long** *window-id* **long** *line-no*)

Returns the text at *line-no*.

**text-window-get-number-of-lines****long** (**text-window-get-number-of-lines** **long** *window-id*)

Returns the number of lines in the text window.

**text-window-load-file****long** (**text-window-load-file** **long** *window-id* **string** *filename*)

Load the file onto the text subwindow, returning 1 for success, 0 for failure.

**text-window-modified****long** (**text-window-modified** **long** *window-id*)

Returns 1 if the text has been modified, 0 otherwise.

**text-window-on-char****long** (**text-window-on-char** **long** *panel-id* **long** *event-id*)

The default implementation of the OnChar callback. Call this to pass intercepted characters through to the text window. Note that under Windows, there seems to be an intermittent GPF bug when using this and then closing the window.

**text-window-paste**



**long (text-window-paste long *window-id*)**

Pastes the text (if any) from the clipboard to the text window.

### **text-window-position-to-char**

**long (text-window-position-to-char long *window-id* long *pos*)**

Returns the character position (starting from zero) for the given index position.

### **text-window-position-to-line**

**long (text-window-position-to-line long *window-id* long *pos*)**

Returns the line number (starting from zero) for the given index position.

### **text-window-remove**

**long (text-window-remove long *window-id* long *start-pos* long *end-pos*)**

Removes the text between the given span selection.

### **text-window-replace**

**long (text-window-replace long *window-id* long *start-pos* long *end-pos* string *text*)**

Replaces the text between the given span selection with the given text.

### **text-window-show-position**

**long (text-window-show-position long *window-id* long *pos*)**

Shows the text at the given index position.

### **text-window-save-file**

**long (text-window-save-file long *window-id* string *filename*)**

Saves the text in the subwindow to the given file, returning 1 for success, 0 for failure.

### **text-window-set-editable**

**long (text-window-set-editable long *window-id* long *editable*)**

Sets the window editable (*editable* is 1) or read-only (*editable* is 0).

**text-window-set-insertion-point**

**long** (**text-window-set-insertion-point** **long** *window-id* **long** *pos*)

Sets the insertion point to the given index position.

**text-window-set-selection**

**long** (**text-window-set-selection** **long** *window-id* **long** *start-pos* **long** *end-pos*)

Sets the selection to the given span of text.

**text-window-write**

**long** (**text-window-write** **long** *window-id* **string** *text*)

Writes the given string into the text window, at the current cursor point.

**text-window-xy-to-position**

**long** (**text-window-xy-to-position** **long** *window-id* **long** *char-position* **long** *line*)

Converts the character and line number (each starting from zero) to a position.

**7.51. Timer**

A timer object can be created to notify the application at regular intervals.

**timer-create**

**long** (**timer-create**)

Creates a timer object. Use `timer-start` to start the timer, and register a `Notify` callback function to receive notification.

**timer-delete**

**long** (**timer-delete** **long** *id*)

Stops and deletes the timer object.

**timer-start**

**long** (**timer-start** **long** *id* **long** *milliseconds*)

Starts the timer, notifying at intervals of duration *milliseconds*.

**timer-stop****long (timer-stop long *id*)**

Stops the timer.

**7.52. Toolbar**

See also *Overview* (page 203)

A toolbar is an array of bitmap buttons, implemented by drawing bitmaps onto a canvas, instead of using the native button implementation.

**toolbar-add-separator****long (toolbar-add-separator long *id*)**

Adds a separator between tools.

**toolbar-add-tool****long (toolbar-add-tool long *id* long *index* long *bitmap-id1* optional long *bitmap-id2* = 0 optional long *is-toggle* = 0 optional double *x* = -1.0 optional double *x* = 1.0 optional long *client-data* = 0 optional string *short-help-string*="" optional string *long-help-string*="")**

Adds a tool to the toolbar. Pass at least one bitmap, the bitmap to be displayed when active and not depressed; and optionally, the bitmap to be displayed when the tool is depressed or toggled. Under Windows, only one bitmap is necessary, and under X, the second bitmap will be created automatically as the inverse of the first button if none is supplied.

You can specify whether the tool is allowed to toggle, and pass a position if you are not going to automatically layout the toolbar with *toolbar-layout*. You can associate client data with the tool.

*short-help-string* is only used by Windows 95 versions of wxCLIPS. The string is used to supply text for a tooltip, a small yellow window that appears as the mouse pointer hovers over the button.

*long-help-string* can be used for longer help strings, such as status line help.

**toolbar-clear-tools****long (toolbar-clear-tools long *id*)**

Clears all the tools from the toolbar.

**toolbar-create****long (toolbar-create long *parent\_id* optional long *x* optional long *y* optional long *width* optional long *height* optional string *style* optional string *orientation* = "wxVERTICAL" optional long *nrows-or-columns* optional long *create-buttons* optional string *name*)**

Creates a toolbar, with a given layout orientation (whether the tools are automatically laid out in

rows or columns) and the number of rows or columns. These parameters are arbitrary if the tools are to be positioned manually and *toolbar-layout* not called.

*style* may be a *bit list* of:

- `wxB_3DBUTTONS`: gives a simple 3D look to the buttons.

*create-buttons* should be 1 (the default) if the toolbar should superimpose the user-supplied buttons onto a larger 3D button. If 0, the tool will be the same size as the button, and the toggle state will be represented by inverting the tool (Windows) or adding a border (X).

Returns the toolbar id if successful, zero otherwise.

*name* gives the toolbar a name that can be retrieved with *window-get-name* (page 176).

Note that absolute tool positioning (or the *toolbar-layout* function) does not work for buttonbars under Windows 95: instead, you can specify the number of rows for the toolbar, and use *toolbar-add-separator* to achieve inter-tool spacing.

## **toolbar-create-tools**

**long (toolbar-create-tools long *id*)**

This should be called when creating Windows 95 buttonbars, after all tools have been added. It adds the tools to the toolbar. You can also call it for non-Windows 95 toolbars and buttonbars, in which case it will have no effect.

## **toolbar-enable-tool**

**long (toolbar-enable-tool long *id* long *tool-id* long *enable*)**

Enables the tool (if *enable* is 1) or disables it (if *enable* is 0).

## **toolbar-get-max-height**

**double (toolbar-get-max-height long *id*)**

Gets the maximum height of the toolbar when it has been automatically laid out.

## **toolbar-get-max-width**

**double (toolbar-get-max-width long *id*)**

Gets the maximum width of the toolbar when it has been automatically laid out.

## **toolbar-get-tool-client-data**

**long (toolbar-get-tool-client-data long *id* long *tool-id*)**

Returns the client data associated with the given tool.

**toolbar-get-tool-enabled****long** (**toolbar-get-tool-enabled** **long** *id* **long** *tool-id*)

Returns 1 if the tool is enabled, 0 otherwise.

**toolbar-get-tool-long-help****string** (**toolbar-get-tool-long-help** **long** *id* **long** *tool-id*)

Returns the long help associated with this tool.

**toolbar-get-tool-short-help****string** (**toolbar-get-tool-short-help** **long** *id* **long** *tool-id*)

Returns the short help associated with this tool.

**toolbar-get-tool-state****long** (**toolbar-get-tool-state** **long** *id* **long** *tool-id*)

Returns the tool state (1 for toggled on, 0 for off).

**toolbar-layout****long** (**toolbar-layout** **long** *id*)

Lays out all the tools if automatic layout is required.

Note that this function does not work for buttonbars under Windows 95: but you can still specify the number of rows for the toolbar.

**toolbar-on-paint****void** (**toolbar-on-paint** **long** *id*)

Calls the default toolbar paint callback. You may wish to call this if you override the default callback.

**toolbar-set-default-size****long** (**toolbar-set-default-size** **long** *id* **long** *width* **long** *height*)

Sets the width and height of tool buttons (Windows only). The default is 24 by 22.

**toolbar-set-margins****long** (**toolbar-set-margins** **long** *id* **long** *x* **long** *y*)

Sets the width and height of the toolbar margins and spacing, if automatic layout is being used.

**toolbar-set-tool-long-help****long** (**toolbar-set-tool-long-help** **long** *id* **long** *tool-id* **string** *help-string*)

Sets the long help associated with this tool.

**toolbar-set-tool-short-help****long** (**toolbar-set-tool-short-help** **long** *id* **long** *tool-id* **string** *help-string*)

Sets the short help associated with this tool.

**toolbar-toggle-tool****long** (**toolbar-toggle-tool** **long** *id* **long** *tool-id* **long** *toggle*)

Toggles the tool on or off.

**7.53. Window**

The window is an 'abstract' class which does not exist in its own right, but is used to access the functionality of classes derived from it. Therefore, please refer to this section when considering other classes.

**window-add-callback****long** (**window-add-callback** **long** *window-id* **word** *event* **word** *function*)

Sets the callback function of a given window (frame, panel, panel item etc.) for the given event, to be the given CLIPS function. See individual window descriptions for details of valid callbacks.

**window-centre****long** (**window-centre** **long** *window-id* **word** *orientation*)

*orientation* may be wxVERTICAL, wxHORIZONTAL or wxBOTH. Centres the window with respect to its parent (or desktop).

**window-close****long** (**window-close** **long** *window-id* **long** *force-close*)

Closes the dialog or frame without immediately deleting the object. The object will be cleaned up in 'idle' processing time. Use of this function instead of deleting the window directly is highly recommended, especially under Motif which is sensitive to frame and dialog deletion.

This function first calls the window's OnClose handler. If OnClose returns FALSE, the close will be vetoed *unless* the *force-close* argument is 1, in which case the deletion will take place anyway.

`window-close` should only be used for frames and dialog boxes.

## **window-delete**

**long (window-delete long *window-id*)**

Deletes a window. See also *window-close* (page 175).

## **window-enable**

**long (window-enable long *window-id* long *enable*)**

If *enable* is 1, enables the window for input. If *enable* is 0, the window is disabled (greyed out in the case of a panel item).

## **window-fit**

**long (window-fit long *window-id*)**

Fits the panel, dialog box or frame around its children.

## **window-get-name**

**string (window-get-name long *window-id*)**

Gets the window's name (the 'name' parameter passed to a window constructor).

## **window-get-next-child**

**long (window-get-next-child long *window-id* long *child-id*)**

If *child-id* is zero, returns the id of the first child window of *window-id*.

If *child-id* is a valid child id, returns the id of the next child window.

Returns -1 if there are no more children.

Example:

```
(bind ?child-id (window-get-next-child ?win-id 0))  
  
(while (neq ?child-id -1)
```

```
(bind ?type (object-get-type ?child-id))  
...  
(bind ?child-id (window-get-next-child ?win-d ?child-id))  
)
```

**window-get-parent**

**long** (**window-get-parent** **long** *window-id*)

Gets the id of the window's parent.

**window-get-x**

**long** (**window-get-x** **long** *window-id*)

Get the x coordinate of the window.

**window-get-y**

**long** (**window-get-y** **long** *window-id*)

Gets the y coordinate of the window.

**window-get-width**

**long** (**window-get-width** **long** *window-id*)

Gets the width of the window.

**window-get-height**

**long** (**window-get-height** **long** *window-id*)

Gets the height of the window.

**window-get-client-width**

**long** (**window-get-client-width** **long** *window-id*)

Gets the client width (space available for child windows) of the window.

**window-get-client-height**

**long** (**window-get-client-height** **long** *window-id*)

Gets the client height (space available for child windows) of the window.



**window-is-shown****long (window-is-shown long window-id)**

Returns 1 if the window is shown, 0 otherwise.

**window-make-modal****long (window-make-modal long window-id long modal)**

*modal* may be 1 to disable all frames and dialog boxes except this one, or 0 to enable all frames and dialogs again.

Has no effect in XView.

**window-popup-menu****long (window-popup-menu long window-id long menu-id double x double y)**

Pops up a menu on the window, at the given position. The menu will be dismissed (but not destroyed) when the user makes a selection.

Note that there is a reliability problem with Motif popup menus; they may not pop up after the first time.

**window-refresh****long (window-refresh long window-id long erase-background=1 long x=-1 long y=-1 long width=-1 long height=-1 )** Refreshes the give window, causing OnPaint to be called. This function should be called in preference to calling an OnPaint handler directly.

*erase-background* controls whether the window background is automatically cleared in the current background colour (1) or not (0). The default is 1.

The last four optional arguments define a rectangle to limit the 'damaged' area. If all arguments are -1, this is taken to mean that the whole window should be refreshed.

**window-remove-callback****long (window-remove-callback long window-id word event)**

Removes the callback function associated with this event.

**window-set-cursor****long (window-set-cursor long window-id long cursor-id)**

Sets the cursor for this window.

**window-set-focus**

**long (window-set-focus long *window-id*)**

Set this window to have the keyboard focus.

**window-set-size**

**long (window-set-size long *window-id* long *x* long *y* long *width* long *height*)**

Sets the position and size of the window.

**window-set-size-hints**

**long (window-set-size-hints long *window-id* long *min-width*=-1 long *min-height*=-1 long *max-width*=-1 long *max-height*=-1 long *inc-width*=-1 long *inc-height*=-1)** Tells the windowing system to restrict the resizing of the frame or dialog box.

*min-width*, *min-height* determine the minimum size of the window.

*max-width*, *max-height* determine the maximum size of the window.

*inc-width*, *inc-height* determine the increments by which the window is sized (Motif only).

-1 values indicate where default values should be used instead of application-specified values.

**window-set-client-size**

**long (window-set-client-size long *window-id* long *width* long *height*)**

Sets the client size (available space for child windows) of the window.

**window-show**

**long (window-show long *window-id* long *show*)**

If *show* is 1, shows the window. If *show* is 0, the window is hidden. If the window is a modal dialog box, *show* = 1 will start the modal loop, and *show* = 0 will terminate the loop (allowing execution to proceed after the first call to *window-show*).

**7.54. Miscellaneous**

This section contains an assortment of useful GUI and other functions.

**batch**

**void (batch string *filename*)**

Executes the given file of CLIPS commands as if from a terminal. Note that full error checking on

construct definitions is not performed; use `load` when checking is required.

## **begin-busy-cursor**

### **void (begin-busy-cursor)**

Starts a 'busy' section of code, putting up an hourglass cursor. Use *end-busy-cursor* (page 181) at the end of the section.

These pairs of calls may be nested for programming convenience.

## **bell**

### **void (bell)**

Rings the system bell.

## **chdir**

### **long (chdir string *directory*)**

Changes to the given directory and returns 1 if successful, 0 otherwise.

## **clean-windows**

### **void (clean-windows)**

Delete all frames and dialog boxes created through CLIPS calls.

## **clear-ide-window**

### **void (clear-ide-window)**

Clears the wxCLIPS development window.

## **clear-resources**

### **long (clear-resources)**

Clears the wxCLIPS resource table. This table is separate from the default resource table that is used by wxCLIPS and other host C++ applications.

See also *load-resource-file* (page 185), *panel-create-from-resource* (page 154), *dialog-box-create-from-resource* (page 122).

## **copy-file**

**long (copy-file string *f1* string *f2*)**

Copies file *f1* to *f2*, returning 1 if successful, 0 otherwise.

**debug-msg****void (debug-msg string *text*)**

Outputs *text* to the debugging stream. Under X, this is the standard error stream. Under Windows, this outputs to the debugger (if present) or any other program that can intercept debug messages, such as Microsoft's DBWIN sample application. This can be useful if you don't have a text window available, and you want the messages to persist after your program has exited, gracefully or otherwise.

**dir-exists****long (dir-exists string *directory*)**

Returns 1 if the directory exists, 0 otherwise.

**end-busy-cursor****void (end-busy-cursor)**

Ends a 'busy' section of code, resetting the cursor to the original for each window. Use *begin-busy-cursor* (page 180) at the start of the section.

These pairs of calls may be nested for programming convenience.

**execute****long (execute string *command* optional long *synchronous* = 0)**

Executes the given system command, either asynchronously (the function returns control immediately) or synchronously (the function returns control when the command terminates). The default is asynchronous execution.

This function should be used in preference to the CLIPS *system* command. Under Windows, it calls WinExec. You cannot call built-in DOS commands (such as *erase*) with this function: you may need to write a batch file instead.

**fact-string-existp****bool (fact-string-existp string *fact*)**

Allows an application to test a fact from within a function. For example:

```
CLIPS> (fact-string-existp "(Example 1)")
CLIPS> FALSE
```

```
CLIPS> (assert (Example 2))
CLIPS> <Fact-0>
CLIPS> (fact-string-existp "(Example 2)")
CLIPS> TRUE
CLIPS> (retract 0)
CLIPS> (fact-string-existp "(Example 2)")
CLIPS> FALSE
```

## file-exists

**long (file-exists string *filename*)**

Returns 1 if the file exists, 0 otherwise.

## file-selector

**string (file-selector optional string *message* optional string *path* optional string *file* optional string *extension* optional string *wildcard* optional long *parent-id* optional string *flags*)**

Pops up a file selector with given (optional) arguments, returning a fully qualified filename or the empty string.

*flags* can be the empty string or a *bit list* of the following:

wxSAVE	Display the Save button instead of the Open button (Windows only).
wxOVERWRITE_PROMPT	Prompts the user when saving if there is already a file of that name (Windows only).
wxOPEN	Display the Open button (Windows only).
wxHIDE_READONLY	Hide the "Open as read-only" checkbox (Windows only).

## find-window-by-label

**long (find-window-by-label string *label* optional long *parent-id*)**

Finds a window with a label or title corresponding to *label*. Optionally pass a parent id from where to start searching.

## find-window-by-name

**long (find-window-by-name string *name* optional long *parent-id*)**

Finds a window with a name corresponding to *name*. Optionally pass a parent id from where to start searching.

## float-to-string

**string (float-to-string double *n*)**

Convert a floating point number to a string.

**get-active-window****long (get-active-window)**

Returns the id of the active window, or -1 if either there is no active window in this application, or the active window has not been created as a wxCLIPS window.

This function only works under MS Windows.

**get-choice****string (get-choice string *message* multifield *choices* optional long *centre-message* optional long *parent-id*)**

Given a message string and a multifield comprising a number of choice strings, pops up a menu for the user to select one item. Returns one of the supplied strings if the user pressed Ok, or the null string if the user pressed Cancel.

A multifield can be created with the CLIPS function *mv-append*, for example:

```
(bind ?choice (get-choice "Choose please"
                          (mv-append "One" "Two" "Three")))
```

If *centre-message* is 1 (the default), the message will be centred on the dialog box. If it is 0, the message will be left-justified. New lines are allowed in the message.

**get-elapsed-time****long (get-elapsed-time optional long *reset-timer* = 1)**

Returns the elapsed time in milliseconds since the last reset, using *start-timer* (page 188) or by passing 1 to this function.

**get-ide-window****long (get-ide-window)**

Gets the id of the wxCLIPS development window (stand-alone wxCLIPS only). If the development window has not been created, zero is returned.

**get-os-version****string (get-os-version)**

Returns a string representing the operating system under which the program is currently running. It is more precise than *get-platform* (page 184). However, be careful about inferring from a value

of wxWIN95 that this version of wxCLIPS is compiled as a Windows 95 application: it may be compiled as a generic WIN32 application.

This may be one of the following (although only a number of these platforms are currently supported).

- wxCURSES: Text-only CURSES platform
- wxXVIEW\_X: Sun's XView OpenLOOK toolkit
- wxMOTIF\_X: OSF Motif 1.x.x
- wxCOSE\_X: OSF Common Desktop Environment
- wxNEXTSTEP: NeXTStep
- wxMACINTOSH: Apple System 7
- wxGEOS: GEOS
- wxOS2\_PM: OS/2 Workplace
- wxWINDOWS: Windows or WfW
- wxPENWINDOWS: Windows for Pen Computing
- wxWINDOWS\_NT: Windows NT
- wxWIN32S: Windows 32S API
- wxWIN95: Windows 95
- wxWIN386: Watcom 32-bit supervisor mode

## **get-platform**

### **string (get-platform)**

Gets a string indicating the current platform the program is running on. Currently one of "XView", "Motif" and "Windows 3.1".

For a more precise notion of current operating system, see *get-os-version* (page 183).

## **get-resource**

### **string (get-resource string section string entry optional string filename)**

Gets the value from the resource file (such as WIN.INI or .Xdefaults, depending on platform). If the filename is omitted, WIN.INI under Windows or .Xdefaults under X will be used.

See also *write-resource* (page 189)

## **get-text-from-user**

### **string (get-text-from-user string message optional string default-value optional long centre-message optional long parent-id)**

Give a message string and a default value, pops up a dialog box prompting the user to enter a string. Returns the input string if the user pressed Ok, or the null string if the user pressed Cancel.

If *centre-message* is 1 (the default), the message will be centred on the dialog box. If it is 0, the message will be left-justified. New lines are allowed in the message.

## load-resource-file

**long (load-resource-file string filename)**

Loads the given .wxx resource file, return 1 if the operation was successful.

See also *clear-resources* (page 180), *panel-create-from-resource* (page 154), *dialog-box-create-from-resource* (page 122).

## long-to-string

**string (long-to-string long value)**

Convert the integer to a string.

## make-metafile-placeable

**long (make-metafile-placeable string filename long min-x long min-y long max-x long max-y optional double scale)**

Given a filename for an existing, valid metafile, makes it into a placeable metafile by prepending a header containing the given bounding box. The bounding box may be obtained from a device context after drawing into it, using the functions *dc-get-min-x*, *dc-get-min-y*, *dc-get-max-x*, and *dc-get-max-y*.

In addition to adding the placeable metafile header, this function adds the equivalent of the following code to the start of the metafile data:

```
SetMapMode(dc, MM_ANISOTROPIC);  
SetWindowOrg(dc, minX, minY);  
SetWindowExt(dc, maxX - minX, maxY - minY);
```

This simulates the *MM\_TEXT* mapping mode, which *wxWindows* assumes.

Placeable metafiles may be imported by many Windows applications, and can be used in RTF (Rich Text Format) files.

*scale* allows the specification of scale for the metafile.

This function is only available under Windows.

See also *metafile-dc* (page 148).

## mci-send-string

**string (mci-send-string string command)**

Sends an MCI (Media Control Interface) string to Windows. Returns an error string if there was an error, or the empty string if there was no error. This allows you to play MIDI and WAV files, for example, and videos if you have an appropriate device driver.



For example:

```
(bind ?err (mci-send-string "play bark.wav"))
(if (neq ?err "") then (printout t "Error: " ?err crlf))
```

The following describes the basic command syntax.

```
load device_name          {file_name}

pause device_name

play device_name          [from position]
                          [to position]
                          [insert | overwrite]

resume device_name

save device_name          [file_name]

seek device_name          {to position | to start | to end}

set device_name           [audio all off
                           | audio all on
                           | audio left off
                           | audio left on
                           | audio right off
                           | audio right on
                           | video off
                           | video on]
                           [door closed | door open]
                           [time format milliseconds | time format ms]

status device_name        {current track
                           | length
                           | length track track_number
                           | mode
                           | number of tracks
                           | position
                           | position track track_number
                           | ready
                           | start position
                           | time format}

stop device name
```

## message-box

**word** (**message-box string** *message* **optional word** *type*  
**optional long** *centre-message* **optional long** *parent-id* **optional string** *title*)

Pops up a dialog box with a message, where the buttons on the dialog box depend on the *type* parameter. This may be OK, OK-CANCEL, YES-NO or YES-NO-CANCEL. The return value is OK, CANCEL, YES or NO.

If *centre-message* is 1 (the default), the message will be centred on the dialog box. If it is 0, the

message will be left-justified. New lines are allowed in the message.

The optional *title* parameter allows the message box title to be changed from the default string 'Message'.

## **mkdir**

**long (mkdir string *directory*)**

Creates the given directory and returns 1 if successful, 0 otherwise.

## **now**

**string (now)**

Returns a string representing the current time and date.

## **read-string**

**string (read-string)**

Read a string (pops up a dialog box).

## **return-result**

**void (return-result any *result*)**

Used by internal C++ functions to get the return value of an arbitrary CLIPS expression.

## **rmdir**

**long (rmdir string *directory*)**

Removes the given directory and returns 1 if successful, 0 otherwise.

## **show-ide-window**

**void (show-ide-window)**

Shows the wxCLIPS development window if it has not already been created (stand-alone wxCLIPS only). This can be useful if starting a CLIPS program from the command line, and you want the development window to be shown before app-on-init has finished. Only likely to work under Windows.

## **set-work-proc**

**void (set-work-proc string *function*)**

Sets the work function, a function with no parameter and no return result, which will be called when the application is otherwise idle. If this is the empty string, the work procedure is cancelled.

(Stand-alone version of wxCLIPS only).

*Note:* this has been found not to work properly on the Windows version, and is not implemented for XView. So probably this is useful only under Motif.

## **sleep**

**long** (**sleep long** *no-secs*)

Makes the process dormant for the given number of seconds. This might be used in a loop involving interprocess communication, for example, to allow time for programs to be loaded. Message processing will take place whilst the process is asleep, so beware of the user interacting with the system during this period.

## **start-timer**

**void** (**start-timer**)

Starts the wxCLIPS stopwatch. You can get elapsed time in milliseconds with *get-elapsed-time* (page 183).

## **string-sort**

**multifield** (**string-sort multifield** *string-list*)

Sorts the given multifield list in ascending alphabetical order. A list may be created using the mv-append CLIPS function.

## **string-to-float**

**double** (**string-to-float string** *value*)

Convert the string to a floating point number.

## **string-to-long**

**long** (**string-to-long string** *value*)

Convert the string to a long integer.

## **string-to-symbol**

**word** (**string-to-symbol string** *value*)

Convert the string to a symbol.

**symbol-to-string****string** (**symbol-to-string** **word** *value*)

Convert the string to a symbol.

**write-resource****long** (**write-resource** **string** *section* **string** *entry* **string** *value* **optional string** *filename*)

Writes the value into the resource file (such as WIN.INI or .Xdefaults, depending on platform). If the filename is omitted, WIN.INI under Windows or .Xdefaults under X will be used.

See also *get-resource* (page 184)

**wxclips-object-exists****long** (**wxclips-object-exists** **long** *id*)

Returns 1 if the given wxCLIPS object exists, 0 otherwise.

**yield****long** (**yield**)

Yields to the windowing system message loop, if appropriate. Normally only of use under Windows, during periods of intensive processing, particularly following window creation or modification. It has no effect under XView or Motif.

## 8. wxCLIPS classes by category

A classification of wxCLIPS classes by category.

### 8.1. Managed windows

There are several types of window that are directly controlled by the window manager (such as MS Windows, or the Motif Window Manager). Frames may contain *subwindows* (page 190), and dialog boxes have their own built-in subwindow similar to a panel.

#### wxCLIPS function groups

- *Frame* (page 123)
- *Dialog box* (page 121)

#### wxCOOL classes

- *wxFrame* (page 48)
- *wxDialogBox* (page 44)

### 8.2. Subwindows

Subwindows should be created as children of frames. The panel subwindow may contain panel items (controls or widgets).

#### wxCLIPS function groups

- *Canvas* (page 96)
- *Grid* (page 129)
- *Panel* (page 154)
- *Text window* (page 167)
- *Toolbar* (page 172)

#### wxCOOL classes

- *wxCanvas* (page 21)
- *wxPanel* (page 66)
- *wxTextWindow* (page 82)
- *wxToolBar* (page 84)

See also *Window* (page 175) and *wxWindow* (page 88).

### 8.3. Panel items

These are widgets (in Motif terminology) or controls (in MS Windows terminology) that can be placed on panels and dialog boxes, with the exception of Menu and MenuBar.

#### wxCLIPS function groups

- *Button* (page 95)
- *CheckBox* (page 100)
- *Choice* (page 100)
- *Gauge* (page 128)
- *GroupBox* (page 136)

- *Item* (page 155)
- *ListBox* (page 141)
- *MultiText* (page 150)
- *Menu* (page 144)
- *MenuBar* (page 145)
- *Message* (page 146)
- *RadioBox* (page 157)
- *Slider* (page 165)
- *Text* (page 166)

#### **wxCool classes**

- *wxButton* (page 20)
- *wxCheckBox* (page 23)
- *wxChoice* (page 23)
- *wxGauge* (page 52)
- *wxGroupBox* (page 53)
- *wxItem* (page 68)
- *wxListBox* (page 55)
- *wxMultiText* (page 65)
- *wxMenu* (page 58)
- *wxMenuBar* (page 60)
- *wxMessage* (page 61)
- *wxRadioBox* (page 71)
- *wxSlider* (page 80)
- *wxText* (page 81)

See also *Window* (page 175) and *wxWindow* (page 88).

### **8.4. Convenience dialogs**

Popup-related special-purpose dialogs, and related functions.

- *file-selector* (page 182)
- *get-choice* (page 183)
- *get-text-from-user* (page 184)
- *message-box* (page 186)
- *begin-busy-cursor* (page 180)
- *end-busy-cursor* (page 181)

### **8.5. Device contexts**

See also *Overview* (page 202)

Device contexts are surfaces that may be drawn on, and provide an abstraction that allows parameterisation of your drawing code by passing different device contexts.

#### **wxCLIPS function groups**

- *wxDC* (page 115)
- *MemoryDC* (page 143)
- *MetaFileDC* (page 148)
- *PostScriptDC* (page 156)

- *PrinterDC* (page 157)
- *Metafile* (page 147)

**wxCool classes**

- *DC* (page 39)
- *wxMemoryDC* (page 58)
- *wxMetaFileDC* (page 62)
- *wxPostScriptDC* (page 69)
- *wxPrinterDC* (page 70)
- *wxMetafile* (page 61)

See also *make-metafile-placeable* (page 185).

**8.6. Graphics device interface**

These classes are related to the Graphics Device Interface, in MS Windows terminology.

**wxCLIPS function groups**

- *Bitmap* (page 93)
- *Brush* (page 95)
- *Cursor* (page 106)
- *Font* (page 123)
- *Icon* (page 138)
- *Pen* (page 156)
- *Colour* (page 103)

**wxCool classes**

- *wxBitmap* (page 18)
- *wxBrush* (page 19)
- *wxCursor* (page 29)
- *wxFont* (page 47)
- *wxIcon* (page 53)
- *wxPen* (page 69)

**8.7. Events**

Some member functions that an application overrides are passed event objects containing information about the event.

**wxCLIPS function groups**

- *CommandEvent* (page 103)
- *Event* (page 122)
- *KeyEvent* (page 140)
- *MouseEvent* (page 148)

**wxCool classes**

- *wxCommandEvent* (page 25)
- *wxEvent* (page 46)
- *wxKeyEvent* (page 54)

- *wxMouseEvent* (page 63)

## 8.8. Interprocess communication

See also *Overview* (page 199)

wxCLIPS provides a simple interprocess communications facilities based on DDE.

### wxCLIPS function groups

- *Client* (page 102)
- *Connection* (page 104)
- *Help* (page 126)
- *Server* (page 165)

### wxCOOL classes

- *wxClient* (page 25)
- *wxConnection* (page 26)
- *wxHelpInstance* (page 51)
- *wxServer* (page 79)

## 8.9. Database classes

See also *Database classes overview* (page 205)

wxCLIPS provides a set of classes for accessing Microsoft's ODBC (Open Database Connectivity) product.

### wxCLIPS function groups

- *Database* (page 107)
- *RecordSet* (page 158)

### wxCOOL classes

- *wxDatabase* (page 31)
- *wxRecordSet* (page 72)

## 8.10. File functions

- *chdir* (page 180)
- *dir-exists* (page 181)
- *file-exists* (page 182)
- *get-resource* (page 184)
- *write-resource* (page 189)

## 8.11. Time-related functions

### Functions

- *Date class* (page 109)
- *Timer class* (page 171)
- *get-elapsed-time* (page 183)



- *start-timer* (page 188)
- *now* (page 187)

#### **wxCLIPS function groups**

- *Date class* (page 109)
- *Timer class* (page 171)

#### **wxCOOL classes**

- *wxDate* (page 33)
- *wxTimer* (page 84)

### **8.12. Noisy functions**

- *bell* (page 180)
- *mci-send-string* (page 185)

### **8.13. Operating system functions**

These functions are related to operating system functionality.

- *execute* (page 181)
- *get-platform* (page 184)
- *get-resource* (page 184)
- *write-resource* (page 189)
- *yield* (page 189)
- *sleep* (page 188)

### **8.14. wxCLIPS environment functions**

These functions are related to the wxCLIPS development environment.

- *app-on-init* (page 93)
- *batch* (page 179)
- *clean-windows* (page 180)
- *debug-msg* (page 181)
- *get-ide-window* (page 183)
- *get-resource* (page 184)
- *show-ide-window* (page 187)

### **8.15. Data functions**

These functions are related to general data manipulation.

- *float-to-string* (page 182)
- *long-to-string* (page 185)
- *read-string* (page 187)
- *string-sort* (page 188)
- *string-to-float* (page 188)
- *string-to-long* (page 188)
- *string-to-symbol* (page 188)
- *symbol-to-string* (page 189)



## 9. Topic overviews

### 9.1. Window styles

Window styles are used to specify alternative behaviour and appearances for windows, when they are created. The symbols are defined in such a way that they can be combined in a 'bit list' using the *bitwise-or* operator, as found in C and C++. In CLIPS, you enclose this bit list in a string. For example:

```
"wxCAPTION | wxMINIMIZE_BOX | wxMINIMIZE_BOX | wxTHICK_FRAME"
```

#### 9.1.1. wxFrame styles

The following styles apply to wxFrame windows.

wxICONIZE	Display the frame iconized (minimized) (Windows only).
wxCAPTION	Puts a caption on the frame (Windows and XView only).
wxDEFAULT_FRAME	Defined as a combination of wxMINIMIZE_BOX, wxMAXIMIZE_BOX, wxTHICK_FRAME, wxSYSTEM_MENU, and wxCAPTION.
wxMDI_CHILD	Specifies a Windows MDI (multiple document interface) child frame.
wxMDI_PARENT	Specifies a Windows MDI (multiple document interface) parent frame.
wxMINIMIZE	Identical to <b>wxICONIZE</b> .
wxMINIMIZE_BOX	Displays a minimize box on the frame (Windows and Motif only).
wxMAXIMIZE	Displays the frame maximized (Windows only).
wxMAXIMIZE_BOX	Displays a maximize box on the frame (Windows and Motif only).
wxSDI	Specifies a normal SDI (single document interface) frame.
wxSTAY_ON_TOP	Stay on top of other windows (Windows only).
wxSYSTEM_MENU	Displays a system menu (Windows and Motif only).
wxTHICK_FRAME	Displays a thick frame around the window (Windows and Motif only).
wxRESIZE_BORDER	Displays a resizable border around the window (Motif only).
wxTINY_CAPTION_HORIZ	Under Windows 3.1, displays a small horizontal caption if USE_ITSY_BITS is set to 1 in wx_setup.h and the Microsoft ItsyBitsy library has been compiled. Use instead of wxCAPTION.
wxTINY_CAPTION_VERT	Under Windows 3.1, displays a small vertical caption if USE_ITSY_BITS is set to 1 in wx_setup.h and the Microsoft ItsyBitsy library has been compiled. Use instead of wxCAPTION.

#### 9.1.2. wxDialogBox styles

The following styles apply to wxDialogBox windows.

wxCAPTION	Puts a caption on the dialog box (Motif only).
wxDEFAULT_DIALOG_STYLE	Equivalent to a combination of wxCAPTION, wxSYSTEM_MENU and wxTHICK_FRAME
wxRESIZE_BORDER	Display a resizable frame around the window (Motif only).
wxSYSTEM_MENU	Display a system menu (Motif only).
wxTHICK_FRAME	Display a thick frame around the window (Motif only).
wxUSER_COLOURS	Under Windows, overrides standard control processing to allow setting of the dialog box background colour.
wxVSCROLL	Give the dialog box a vertical scrollbar (XView only).

### 9.1.3. wxItem styles

The following styles apply to all *wxItem* (page 68) derived windows.

`wxHORIZONTAL_LABEL`      The item will be created with a horizontal label.  
`wxVERTICAL_LABEL`      The item will be created with a vertical label.  
`wxFIXED_LENGTH`      Allows the values of a column of items to be left-aligned. Create an item with this style, and pad out your labels with spaces to the same length. The item labels will initially be created with a string of identical characters, positioning all the values at the same x-position. Then the real label is restored.

### 9.1.4. wxButton styles

There are no styles specific to *wxButton* (page 20).

### 9.1.5. wxGauge styles

The following styles apply to *wxGauge* (page 52) items.

`wxGA_HORIZONTAL`      The item will be created as a horizontal gauge.  
`wxGA_VERTICAL`      The item will be created as a vertical gauge.  
`wxGA_PROGRESSBAR`      Under Windows 95, the item will be created as a horizontal progress bar.

### 9.1.6. wxGroupBox styles

There are no styles specific to *wxGroupBox* (page 53).

### 9.1.7. wxListBox styles

The following styles apply to *wxListBox* (page 55) items.

`wxNEEDED_SB`      Create scrollbars if needed.  
`wxLB_NEEDED_SB`      Same as `wxNEEDED_SB`.  
`wxALWAYS_SB`      Create scrollbars immediately.  
`wxLB_ALWAYS_SB`      Same as `wxALWAYS_SB`.  
`wxLB_SINGLE`      Single-selection list.  
`wxLB_MULTIPLE`      Multiple-selection list.  
`wxLB_EXTENDED`      Extended-selection list (Motif only).  
`wxHSCROLL`      Create horizontal scrollbar if contents are too wide (Windows only).

### 9.1.8. wxMessage styles

There are no styles specific to *wxMessage* (page 61).

### 9.1.9. wxRadioBox

The following styles apply to *wxRadioBox* (page 71) items.

wxVERTICAL    Lays the radiobox out in columns.  
wxHORIZONTAL    Lays the radiobox out in rows.

### 9.1.10. wxSlider styles

The following styles apply to *wxSlider* (page 80) items.

wxHORIZONTAL    The item will be created as a horizontal slider.  
wxVERTICAL    The item will be created as a vertical slider.

### 9.1.11. wxText/wxMultiText styles

The following styles apply to *wxText* (page 81) and *wxMultiText* (page 65) items.

wxTE\_PROCESS\_ENTER    The callback function will receive the event  
                         wxEVENT\_TYPE\_TEXT\_ENTER\_COMMAND. Note that this will break tab  
                         traversal for this panel item under Windows. Single-line text only.  
wxTE\_PASSWORD    The text will be echoed as asterisks. Single-line text only.  
wxTE\_READONLY    The text will not be user-editable.  
wxHSCROLL    A horizontal scrollbar will be displayed. If wxHSCROLL is omitted, only a vertical  
                 scrollbar is displayed, and lines will be wrapped. This parameter is ignored  
                 under XView. Multi-line text only.

### 9.1.12. wxTextWindow styles

The following styles apply to *wxTextWindow* (page 82) objects.

wxBORDER    Use this style to draw a thin border in Windows 3 (non-native implementation  
                 only).  
wxNATIVE\_IMPL    Use this style to allow editing under Windows 3.1, albeit with a 64K  
                 limitation.  
wxREADONLY    Use this style to disable editing.  
wxHSCROLL    Use this style to enable a horizontal scrollbar, or leave it out to allow line  
                 wrapping. Windows and Motif only.

### 9.1.13. wxPanel styles

The following styles apply to *wxPanel* (page 66) windows.

wxBORDER    Draws a thin border around the panel.  
wxUSER\_COLOURS    Under Windows, overrides standard control processing to allow setting of  
                 the panel background colour.  
wxVSCROLL    Gives the dialog box a vertical scrollbar (XView only).

### 9.1.14. wxCanvas styles

The following styles apply to *wxCanvas* (page 21) windows.

**wxBORDER** Gives the canvas a thin border (Windows 3 and Motif only).  
**wxRETAINED** Gives the canvas a wxWindows-implemented backing store, making repainting much faster but at a potentially costly memory premium (XView and Motif only).

### 9.1.15. wxToolBar styles

The following styles apply to *wxToolBar* (page 84) objects.

**wxBTB\_3DBUTTONS** Gives a 3D look to the buttons, but not to the same extent as *wxButtonBar*.

## 9.2. Interprocess communication overview

wxCLIP function groups: *Server* (page 165), *Connection* (page 104), *Client* (page 102).

wxCOOL classes: *wxServer* (page 79), *wxConnection* (page 26), *wxClient* (page 25).

The following describes how wxCLIPS implements DDE. The following three classes are central.

1. **Client.** This represents the client application, and is used only within a client program.
2. **Server.** This represents the server application, and is used only within a server program.
3. **Connection.** This represents the connection from the current client or server to the other application (server or client), and can be used in both server and client programs. Most DDE transactions operate on this object.

Messages between applications are usually identified by three variables: connection object, topic name and item name. A data string is a fourth element of some messages. To create a connection (a conversation in Windows parlance), the client application sends the message *client-make-connection* to the client object, with a string service name to identify the server and a topic name to identify the topic for the duration of the connection. Under UNIX, the service name must contain an integer port identifier.

The server then responds and either vetos the connection or allows it. If allowed, a connection object is created which persists until the connection is closed. The connection object is then used for subsequent messages between client and server.

To create a working server, the programmer must:

1. Create a server object, giving it a service name.
2. Register the callback *OnAcceptConnection* for accepting or rejecting a connection, on the basis of the topic argument.
3. Create a Connection object.
4. Provide callbacks for various messages that are sent to the server side of a Connection.

To create a working client, the programmer must:

1. Create a client object.
2. Create a connection object using *client-make-connection* (page 102).
3. Provide callbacks for various messages that are sent to the client side of a Connection.
4. Use the Connection functions to send messages to the server.

### 9.2.1. Data transfer

These are the ways that data can be transferred from one application to another.

- **Execute:** the client calls the server with a data string representing a command to be executed. This succeeds or fails, depending on the server's willingness to answer. If the client wants to find the result of the Execute command other than success or failure, it has to explicitly call Request.
- **Request:** the client asks the server for a particular data string associated with a given item string. If the server is unwilling to reply, the return value is NULL. Otherwise, the return value is a string (actually a pointer to the connection buffer, so it should not be deallocated by the application).
- **Poke:** The client sends a data string associated with an item string directly to the server. This succeeds or fails.
- **Advise:** The client asks to be advised of any change in data associated with a particular item. If the server agrees, the server will send an OnAdvise message to the client along with the item and data.

The default data type is wxCF\_TEXT (ASCII text), and the default data size is the length of the null-terminated string. Windows-specific data types could also be used on the PC.

### 9.2.2. Connection overview

See also *Interprocess communication overview* (page 199)

A connection object has no creation function, since it is implicitly created when a connection is requested (one object at each side of the connection).

A connection object id is used for initiating DDE commands and requests using functions such as connection-execute, and it also has event handlers associated with it to respond to commands from the other side of the connection.

The callbacks you can define for a connection (using *window-add-callback* (page 175)) are as follows.

- OnAdvise** Called when an OnAdvise message is received by the client in response to a server-side connection-advise call. The function should take arguments: connection id, OnAdvise, topic string, item name string, data string. The function should return 1 if successful, 0 otherwise. The data string is what the server is passing to the client.
- OnExecute** Called when an OnExecute message is received by the server in response to a client-side connection-execute call. The function should take arguments: connection id, OnExecute, topic string, dummy item, data string. The function should return 1 if successful, 0 otherwise.
- OnPoke** Called when an OnPoke message is received by the server in response to a client-side connection-poke call. The function should take arguments: connection id, OnPoke, topic string, item name, data string. The function should return 1 if successful, 0 otherwise.

**OnRequest** Called when an OnRequest message is received by the server in response to a client-side connection-request call. The function should take arguments: connection id, OnRequest, topic string, item name, data string. The function should return the data being requested, or the empty string if none. otherwise.

**OnStartAdvise** Called when an OnStartAdvise message is received by the server in response to a client-side connection-start-advise call. The function should take arguments: connection id, OnStartAdvise, topic string, item name, dummy data. The function should return 1 if successful, 0 otherwise.

**OnStopAdvise** Called when an OnStopAdvise message is received by the server in response to a client-side connection-start-advise call. The function should take arguments: connection id, OnStopAdvise, topic string, item name, dummy data. The function should return 1 if successful, 0 otherwise.

### 9.2.3. Examples

See the sample programs ddeserv.clp, ddeclien.clp in the examples directory. Run the server, then the client (you'll have to copy wxclips.exe to wxclips2.exe to run two copies simultaneously).

The sample ddetest.clp shows a simple example of accessing the Program Manager using DDE (Windows only).

```
;;; Demo of DDE functions: chatting to PROGMAN
;;;

(defglobal ?*progman-server* = 0)
(defglobal ?*progman-server-name* = "PROGMAN")
(defglobal ?*progman-host-name* = "none")
(defglobal ?*progman-topic-name* = "PROGMAN")
(defglobal ?*progman-client* = 0)
(defglobal ?*progman-connection* = 0)

;;; Convert a multifield list of strings to one string
(defun many-strings-to-one ($?strings)
  (bind ?counter 1)
  (bind ?string "")
  (while (<= ?counter (length $?strings)) do
    (bind ?string (str-cat ?string (nth ?counter $?strings)))
    (bind ?counter (+ ?counter 1))
  )
  (return ?string)
)

(defun progman-demo ()
  ;; Get a group name from the user
  (bind ?new-group-name (get-text-from-user "New PROGMAN group name"))
  (if (neq ?new-group-name "") then
    ;; Form create group command
    (bind ?command (many-strings-to-one (mv-append "[CreateGroup(" ?new-group-name ")"])))

    ;; Construct a client object
    (bind ?*progman-client* (client-create))

    ;; Construct a connection object
```



```
(bind ?*progman-connection* (client-make-connection
                             ?*progman-client* ?*progman-host-name*
                             ?*progman-server-name* ?*progman-topic-name*))

;; Execute a command to create a group
(bind ?exe (connection-execute ?*progman-connection* ?command))

;; Request a list of groups
(bind ?req (connection-request ?*progman-connection* "PROGMAN"))
(format t "%nProgram Manager Groups:%n")
(format t "%s%n%n" ?req)

;; Disconnect
(connection-disconnect ?*progman-connection*)
)
)

;;; Automatically called when running application from command line
;;; e.g. wxclips -start -clips ddetest.clp
;;; Also runnable from the Application: Run application.
(defun app-on-init ()
  (progman-demo)
)
```

### 9.3. Device context overview

wxCLIPS function groups: *DC* (page 115), *PostScriptDC* (page 156), *MetaFileDC* (page 148), *MemoryDC* (page 143), *PrinterDC* (page 157)

wxCOOL classes: *wxDC* (page 39), *wxPostScriptDC* (page 69), *wxMetaFileDC* (page 62), *wxMemoryDC* (page 58), *wxPrinterDC* (page 70)

A device context is an abstraction of all the devices that can be drawn onto, such as PostScript file, canvas, printer, metafile, and bitmap. Instead of drawing directly on one of these devices, the application programmer can write a function that writes to a device context, and then pass any device context to that function. The most frequently used device context is probably the canvas device context. This cannot be created by an application but can be retrieved from a *canvas* (page 96) by calling *canvas-get-dc* (page 97).

At present, wxCLIPS supports the canvas, memory, PostScript, Windows printer and Windows metafile device contexts.

When writing code to draw into a device context, use a device context variable as a parameter whenever possible, to allow the most general use of your drawing code. You can then pass a device context object of any derived type.

### 9.4. Dialog box overview

Function group/class: *DialogBox* (page 121)/*wxDialogBox* (page 44)

A dialog box is similar to a panel, in that it is a window which can be used for placing panel items, with the following exceptions:

1. A surrounding frame is implicitly created.
2. Extra functionality is automatically given to the dialog box, such as tabbing between items (currently Windows only).

3. If the dialog box is *modal*, the calling program is blocked until the dialog box is dismissed.

Under XView, some panel items may display incorrectly in a modal dialog, and two modal dialogs may not be open simultaneously. This can be corrected using a `wxWindows` patch.

Under implementations that permit it, Dialog box inherits from Canvas via Panel, and has a Panel DC that the application can draw on.

The panel device context associated with Dialog box behaves slightly differently than for a panel or canvas: drawing to it *requires* enclosing code in `dc-begin-drawing`, `dc-end-drawing` calls. This is because under Windows, dialog box device contexts are not 'retained' and settings would be lost if the device context were retrieved and released for each drawing operations.

See *Miscellaneous* (page 179) for convenience dialog functions which avoid the need to create a dialog box and individual items.

The following callbacks are valid for the dialog box class:

- OnCommand** Called with a panel identifier, an item identifier and a command event identifier when a command event is received by a panel item that does not have an associated callback. If you have created a panel or dialog box from a resource, you will need to intercept `OnCommand`.
- OnClose** The function is called with the window identifier. If the callback returns 1 and the function was called by the window manager, the window is automatically deleted. A return value of 0 forbids automatic deletion.
- OnEvent** Called with a dialog box identifier and a *mouse event* (page 148) identifier. This can only be guaranteed only when the dialog box is in user edit mode (to be implemented).
- OnPaint** Called with a dialog box identifier when the dialog box receives a repaint event from the window manager.
- OnSize** The function is called with the dialog box identifier, width and height.

See also *Panel* (page 154) and *Window* (page 175) for inherited member functions.

## 9.5. Toolbar overview

Function group/class: *Toolbar* (page 172)/*wxToolBar* (page 84)

A toolbar is an array of bitmap buttons, implemented by drawing bitmaps onto a canvas, instead of using the native button implementation. Since the toolbar inherits from canvas, you can use all canvas functions on a toolbar object.

Each tool can be specified as a normal button, on/off toggle, and disabled or enabled. Tool layout is automatic or manual. Left click and right click events may be intercepted, using `OnLeftClick` and `OnRightClick` callbacks. The `OnMouseEnter` callback is used to update the status line (for example) with help text as the mouse moves over the tools. See *window-add-callback* (page 175) for details on these callbacks.

Normal subwindow geometry considerations are applicable (i.e., in a frame with more than one subwindow, you must resize the subwindows when you receive an `OnSize` event from the frame). The exception is for Multiple Document Interface (MDI) frames under Windows, where you must call *frame-set-tool-bar* to associate the toolbar with the MDI client window, and after initializing the toolbar height, further resizing is unnecessary.

Toolbars are often displayed as a horizontal strip under the menubar, or in a floating frame. If you

wish to draw a border for the toolbar, you must intercept the toolbar's `OnPaint` handler. In this overridden callback, you must first call the toolbar's *toolbar-on-paint* function to draw all the tools, and then draw the border onto the toolbar canvas.

*Note* that under Windows, you must supply bitmaps that are 16 pixels wide and 15 pixels high: they will be placed on a tool button that is 24 by 22 pixels. If you wish to supply bitmaps of a different size, you must call *toolbar-set-default-size* to set the overall tool button size (as opposed to the bitmap size), or use the toolbar in non-button-creation mode by supplying an extra argument to *toolbar-create* to disable this functionality.

*Note also* that in some circumstances, especially for the WIN32 version of wxCLIPS, there are problems with the buttonbar (the symptoms are bitmaps scrambled randomly). If this happens, revert to the normal toolbar by passing 0 in the create-buttons argument to *toolbar-create*, or download a Windows 95 version of wxCLIPS.

Under X, tool buttons are the same size as the supplied button and there is no need to call *toolbar-set-default-size*.

**Tip:** in circumstances where you might think of using drag and drop, which is not currently implemented in wxWindows or wxCLIPS, you can use a toolbar to select 'modes' of operation which change the cursor in a subwindow. Intercept left-click in the subwindow to place an object or perform some operation.

Canvas callbacks apply, plus:

- OnLeftClick** The function is called with the toolbar identifier, tool index, and an integer which is 1 if the tool is being toggled on, or zero otherwise. If this is a toggle tool, return 1 to allow the toggle to take place, or 0 otherwise.
- OnRightClick** The function is called with the toolbar identifier, tool index, and x and y floating point parameters indicating the position of the click. No value need be returned.
- OnMouseEnter** The function is called with the tool index, whenever the mouse goes into a tool, or out of all tools. In the latter case, the tool index is -1. No value need be returned.

### 9.5.1. Differences in toolbar types

Different toolbar code kicks in according to the platform, and the arguments given to *tool-bar-create*.

1. If *create-buttons* is 0, then the bog-standard wxToolBar class from wxWindows is used: no 3D effect. This works across all supported wxCLIPS platforms. Layout can either be automated, or tools must be placed at absolute coordinates.
2. If *create-buttons* is 1 and the platform is Windows 3.1 or generic WIN32 (not Windows 95), then the buttons will be 3D effect using the wxButtonBar class. On WIN32 toggle tools will not work. On UNIX, the standard wxToolBar code will be used instead of wxButtonBar. Again, layout is automatic or absolute.
3. If *create-buttons* is 1 and the Windows 95 version of wxCLIPS is being used (not just the WIN32 version running on Windows 95), then the toolbar common control is used, supporting tooltips. However, layout is different: you must specify wxVERTICAL for layout orientation, plus the number of rows (usually 1), and you need to use *toolbar-add-separator* to get spaces between tools. You cannot place tools at absolute coordinates or use the *toolbar-layout* function. You must also call *toolbar-create-tools* after adding tools. Device context painting is restricted and no events may be intercepted for the toolbar except `OnLeftClick` and `OnMouseEnter`.

*Note:* under Windows 95, a wxButtonBar cannot be moved to any position other than the top-left

of the frame.

## 9.6. Database classes overview

wxCLIPS function groups: *Database* (page 107), *Recordset* (page 158)

wxCOOL classes: *wxDatabase* (page 31), *wxRecordSet* (page 72)

**IMPORTANT NOTE:** The ODBC classes are a preliminary release and incomplete. Please take this into account when using them. Feedback and bug fixes are appreciated, as always. The classes are being developed by Olaf Klein (oklein@smallo.ruhr.de) and Patrick Halke (patrick@zaphod.ruhr.de).

wxCLIPS provides a set of classes for accessing a subset of Microsoft's ODBC (Open Database Connectivity) product. Currently, this wrapper is available under MS Windows only, although ODBC may appear on other platforms, and a generic or product-specific SQL emulator for the ODBC classes may be provided in wxWindows at a later date.

ODBC presents a unified API (Application Programmer's Interface) to a wide variety of databases, by interfacing indirectly to each database or file via an ODBC driver. The language for most of the database operations is SQL, so you need to learn a small amount of SQL as well as the wxCLIPS ODBC wrapper API. Even though the databases may not be SQL-based, the ODBC drivers translate SQL into appropriate operations for the database or file: even text files have rudimentary ODBC support, along with dBASE, Access, Excel and other file formats.

The run-time files for ODBC are bundled with many existing database packages, including MS Office.

The minimum you need to distribute with your application is `odbc.dll`, which must go in the Windows system directory. For the application to function correctly, ODBC drivers must be installed on the user's machine. If you do not use the database classes, `odbc.dll` will be loaded but not called (so ODBC does not need to be setup fully if no ODBC calls will be made).

A sample is distributed with wxCLIPS in `examples/odbc`.

### 9.6.1. Procedures for writing an ODBC application

You first need to create a Database object. If you want to get information from the ODBC manager instead of from a particular database (for example using *recordset-get-data-sources* (page 160)), then you do not need to call *database-open* (page 109). If you do wish to connect to a datasource, then call *database-open*. You can reuse your Database object, calling *database-close* and *database-open* multiple times.

Then, create a Recordset object for retrieving or sending information. For ODBC manager information retrieval, you can create it as a dynaset (retrieve the information as needed) or a snapshot (get all the data at once). If you are going to call *recordset-execute-sql* (page 158), you need to create it as a snapshot. Dynaset mode is not yet implemented for user data.

Having called a function such as *recordset-execute-sql* or *recordset-get-data-sources*, you may have a number of records associated with the recordset, if appropriate to the operation. You can now retrieve information such as the number of records retrieved and the actual data itself. Use functions such as *recordset-get-int-data* (page 161) or *recordset-get-char-data* (page 159) to get the data, passing a column index or name. The data returned will be for the current record. To move around the records, use *recordset-move-next* (page 164), *recordset-move-prev* (page 164)

and associated functions.

You can use the same recordset for multiple operations, or delete the recordset and create a new one.

Note that when you delete a Database, any associated recordsets also get deleted, so beware of holding onto invalid pointers.

### 9.6.2. Examples

Here's an example of a function that updates a value in a database.

```
;;; Function for updating a field in a record in the incident.dbf demo
;;; file.
;;; E.g. (demo-update-integer "BD34" "X" 999)
;;; The key is the ASSET column, BD34 in the example. Record(s)
matching
;;; this key will be changed.
;;; "X" is the name of the column to be updated.
;;; 999 is a value to replace the current value.
;;;
;;; You must have previously registered the file incident.dbf
;;; with ODBC (e.g. from the control panel), with the source
;;; name "wxCLIPS demo". You can check if the file has changed
;;; by using Microsoft Query.

(defun demo-update-integer (?asset ?col ?value)
  (bind ?database (database-create))

  ;; Open data source
  (if (eq 0 (database-open ?database "wxCLIPS demo")) then
    (bind ?msg (database-get-error-message ?database))
    (printout t ?msg crlf)
    (return 0)
  )

  ;; Create a recordset
  (bind ?recordset (recordset-create ?database "wxOPEN_TYPE_SNAPSHOT"))

  ;; Construct an SQL statement
  (bind ?sql (str-cat "UPDATE Incident SET " ?col " = " ?value " WHERE
ASSET = ' " ?asset "'))
  (printout t ?sql crlf)

  ;; Execute the SQL.
  (if (eq 0 (recordset-execute-sql ?recordset ?sql)) then

    (bind ?msg (database-get-error-message ?database))
    (printout t ?msg crlf)
    (return 0)
  )

  (recordset-delete ?recordset)
  (database-close ?database)
  (database-delete ?database))
```

```
(return 1)
)
```

The next example gets a value from a particular field of a record.

```
;;; Function for returning the value of an integer field.
;;; E.g. (demo-get-integer "BD34" "X")
;;; The key is the ASSET column, BD34 in the example. The first record
matching
;;; this key will be returned.
;;; "X" is the name of the column whose value is to be returned.

(deffunction demo-get-integer (?asset ?col)
  (bind ?database (database-create))

  ;; Open data source
  (if (eq 0 (database-open ?database "wxCLIPS demo")) then
    (bind ?msg (database-get-error-message ?database))
    (printout t ?msg crlf)
    (return 0)
  )

  ;; Create a recordset
  (bind ?recordset (recordset-create ?database "wxOPEN_TYPE_SNAPSHOT"))

  ;; Construct an SQL statement
  (bind ?sql (str-cat "SELECT * FROM Incident WHERE ASSET = '" ?asset
    "'"))
  (printout t ?sql crlf)

  ;; Execute the SQL.
  (if (eq 0 (recordset-execute-sql ?recordset ?sql)) then

    (bind ?msg (database-get-error-message ?database))
    (printout t ?msg crlf)
    (return 0)
  )

  ;; Get the relevant field of the first record
  (bind ?data (recordset-get-int-data ?recordset ?col))

  (recordset-delete ?recordset)
  (database-close ?database)
  (database-delete ?database)
  (return ?data)
)
```

You can find out all the source names available to you with the following code.

```
(bind ?*database* (database-create))
(bind ?*recordset* (recordset-create ?*database*
"wxOPEN_TYPE_SNAPSHOT"))

;;; Get the list of currently-defined ODBC sources
(if (eq 0 (recordset-get-data-sources ?*recordset*)) then

  (show-database-error) else
```

```
;;; Loop through all the source names (one per record)
(bind ?cont 1)
(while (eq ?cont 1)
  ;;; The source name is at the first column (0) in the record
  (bind ?data (recordset-get-char-data ?*recordset* 0))
  (list-box-append ?*sources-listbox* ?data)
  (bind ?cont (recordset-move-next ?*recordset*)))
)
```

### 9.6.3. Database overview

See also *Database classes overview* (page 205)

Function group/class: *Database* (page 107)/*wxDatabase* (page 31)

Every database object represents an ODBC connection. To do anything useful with a database object you need to create a Recordset object. All you can do with Database is opening/closing connections and getting some info about it (users, passwords, and so on).

### 9.6.4. Recordset overview

See also *Database classes overview* (page 205)

Function group/class: *Recordset* (page 158)/*wxRecordSet* (page 72)

Each Recordset represents a database query. You can make multiple queries at a time by using multiple Recordsets with a Database or you can make your queries in sequential order using the same Recordset.

If Recordset is of the type `wxOPEN_TYPE_DYNASET`, there will be only one field for each column, which will be updated every time you call functions like `recordset-move` or `recordset-goto`. If Recordset is of the type `wxOPEN_TYPE_SNAPSHOT`, all records returned by an ODBC function will be loaded at once.

### 9.6.5. ODBC SQL data types

See also *Database classes overview* (page 205)

These are the data types supported in ODBC SQL. Note that there are other, extended level conformance types, not currently supported in wxCLIPS.

CHAR( <i>n</i> )	A character string of fixed length <i>n</i> .
VARCHAR( <i>n</i> )	A varying length character string of maximum length <i>n</i> .
LONG VARCHAR( <i>n</i> )	A varying length character string: equivalent to VARCHAR for the purposes of ODBC.
DECIMAL( <i>p</i> , <i>s</i> )	An exact numeric of precision <i>p</i> and scale <i>s</i> .
NUMERIC( <i>p</i> , <i>s</i> )	Same as DECIMAL.
SMALLINT	A 2 byte integer.
INTEGER	A 4 byte integer.
REAL	A 4 byte floating point number.

FLOAT            An 8 byte floating point number.  
DOUBLE PRECISION   Same as FLOAT.

These data types correspond to the following ODBC identifiers:

SQL\_CHAR        A character string of fixed length.  
SQL\_VARCHAR     A varying length character string.  
SQL\_DECIMAL     An exact numeric.  
SQL\_NUMERIC     Same as SQL\_DECIMAL.  
SQL\_SMALLINT    A 2 byte integer.  
SQL\_INTEGER     A 4 byte integer.  
SQL\_REAL        A 4 byte floating point number.  
SQL\_FLOAT       An 8 byte floating point number.  
SQL\_DOUBLE      Same as SQL\_FLOAT.

### 9.6.6. A selection of SQL commands

See also *Database classes overview* (page 205)

The following is a very brief description of some common SQL commands, with examples.

#### 9.6.6.1. Create

Creates a table.

Example:

```
CREATE TABLE Book
(
  BookNumber      INTEGER      PRIMARY KEY
, CategoryCode   CHAR(2)      DEFAULT 'RO' NOT NULL
, Title          VARCHAR(100) UNIQUE
, NumberOfPages  SMALLINT
, RetailPriceAmount NUMERIC(5,2)
)
```

#### 9.6.6.2. Insert

Inserts records into a table.

Example:

```
INSERT INTO Book
(BookNumber, CategoryCode, Title)
VALUES(5, 'HR', 'The Lark Ascending')
```

#### 9.6.6.3. Select

The Select operation retrieves rows and columns from a table. The criteria for selection and the columns returned may be specified.



Examples:

```
SELECT * FROM Book
```

Selects all rows and columns from table Book.

```
SELECT Title, RetailPriceAmount FROM Book WHERE RetailPriceAmount > 20.0
```

Selects columns Title and RetailPriceAmount from table Book, returning only the rows that match the WHERE clause.

```
SELECT * FROM Book WHERE CatCode = 'LL' OR CatCode = 'RR'
```

Selects all columns from table Book, returning only the rows that match the WHERE clause.

```
SELECT * FROM Book WHERE CatCode IS NULL
```

Selects all columns from table Book, returning only rows where the CatCode column is NULL.

```
SELECT * FROM Book ORDER BY Title
```

Selects all columns from table Book, ordering by Title, in ascending order. To specify descending order, add DESC after the ORDER BY Title clause.

```
SELECT Title FROM Book WHERE RetailPriceAmount >= 20.0 AND RetailPriceAmount <= 35.0
```

Selects records where RetailPriceAmount conforms to the WHERE expression.

#### 9.6.6.4. Update

Updates records in a table.

Example:

```
UPDATE Incident SET X = 123 WHERE ASSET = 'BD34'
```

This example sets a field in column 'X' to the number 123, for the record where the column ASSET has the value 'BD34'.

### 9.7. Grid overview

Function group/class: *Grid* (page 129)

The grid class is a window designed for displaying data in tabular format. Possible uses include:

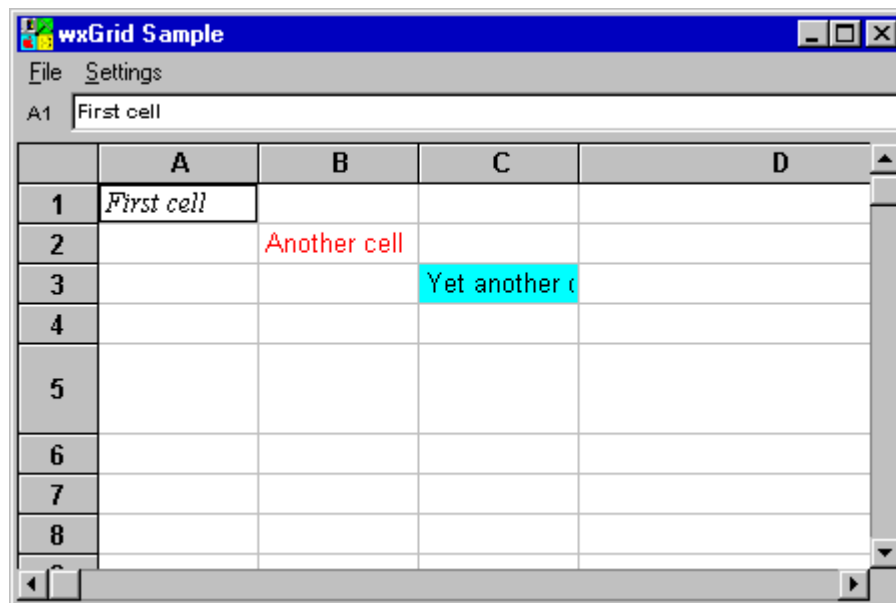
- Displaying database tables;
- building spreadsheet applications;
- displaying files and their attributes;
- use as a more sophisticated listbox where different fonts and colours are required.

This manual currently describes the version of Grid that operates under Windows, implementing

using mostly generic wxWindows code. It is intended to provide a similar API for Motif using the public domain Xbae matrix widget, included in the wxGrid distribution. Work needs to be done to wrap the Xbae functionality in a similar API.

### 9.7.1. The appearance and behaviour of a grid

The following screenshot shows the initial appearance of the sample grid application.



The Grid class is a panel that provides a text editing area, and a grid with scrollbars. The grid has horizontal and vertical label areas whose colours may be changed independently from the cell area. The text editing area, and the label areas, may be switched off if desired.

The user navigates the grid using the mouse to click on cells and scroll around the virtual grid area (no keyboard navigation is possible as yet). If the edit control is enabled, it always has the focus for the currently selected cell and the user can type into it. The text in the edit control will be reflected in the currently selected cell.

If the row and column label areas are enabled, the user can drag on the label divisions to resize a row or column.

The sample application allows the user to change various aspects of the grid using the Grid API. These include:

- Changing the background and foreground colour of labels and cells;
- toggling row and column label areas on and off independently;
- toggling the edit control on and off;
- toggling the light grey cell dividers on and off;
- changing cell alignment.

There are various other aspects that can be controlled via the API, including changing individual cell font and colour properties.

You need to call `grid-create-grid` before there are any cells in the grid.

All row and column positions start from zero, and dimensions are in pixels.

If you make changes to row or column dimensions, call `grid-update-dimensions` and then `grid-adjust-scrollbars`. If you make changes to the grid appearance (such as a change of cell background colour or font), call `window-refresh` for the changes to be shown.

### 9.7.2. Example

The following is an example of using the grid functionality.

```
;;; grid.clp
;;; grid test
;;; Load using -clips <file> on the command line or using the Batch
;;; or Load commands from the CLIPS development window; type
;;; (app-on-init) to start.

(defglobal ?*main-frame* = 0)
(defglobal ?*grid* = 0)

(deffunction on-close (?frame)
  (format t "Closing frame.%n")
  (bind ?*grid* 0)
  1)

(deffunction on-activate (?frame ?active)
  (if (> ?*grid* 0) then (grid-on-activate ?*grid* ?active))
)

(deffunction on-menu-command (?frame ?id)
  (switch ?id
    (case 200 then (message-box "CLIPS for wxWindows Demo
by Julian Smart (c) 1993" wxOK 1 0 "About wxWindows CLIPS Demo"))
    (case 3 then (if (on-close ?frame) then (window-delete ?frame)))
  )
)

;;; Test program to create a frame
(deffunction app-on-init ()
  (unwatch all)

  (bind ?*main-frame* (frame-create 0 "wxCLIPS Grid Test" -1 -1 400
300))

  (window-add-callback ?*main-frame* OnClose on-close)
  (window-add-callback ?*main-frame* OnMenuCommand on-menu-command)
  (window-add-callback ?*main-frame* OnActivate on-activate)

  ;;; Make a menu bar
  (bind ?file-menu (menu-create))
  (menu-append ?file-menu 3 "&Quit")

  (bind ?menu-bar (menu-bar-create))
  (menu-bar-append ?menu-bar ?file-menu "&File")
```

```
(frame-set-menu-bar ?*main-frame* ?menu-bar)

;;; Make a grid
(bind ?*grid* (grid-create ?*main-frame* 0 0 400 300))
(grid-create-grid ?*grid* 10 8)
(grid-set-column-width ?*grid* 3 200)
(grid-set-row-height ?*grid* 4 45)
(grid-set-cell-value ?*grid* "First cell" 0 0)
(grid-set-cell-value ?*grid* "Another cell" 1 1)
(grid-set-cell-value ?*grid* "Yet another cell" 2 2)
(grid-set-cell-text-font ?*grid* (font-create 12 wxROMAN wxITALIC
wxNORMAL 0) 0 0)
(bind ?red (colour-create RED))
(grid-set-cell-text-colour ?*grid* ?red 1 1)
(bind ?cyan (colour-create CYAN))
(grid-set-cell-background-colour ?*grid* ?cyan 2 2)
(grid-update-dimensions ?*grid*)

(window-centre ?*main-frame* wxBOTH)

(window-show ?*main-frame* 1)

?*main-frame*)
```

## 9.8. wxCOOL overview

### 9.8.1. What is wxCOOL?

Up until July 1995, wxCLIPS functionality was conceptually object-oriented, but solely implemented using CLIPS functions. Since the only way to couple CLIPS to C or C++ programs is by defining user functions, the functional route is a prerequisite. wxCOOL is a set of CLIPS classes built on top of the user functions, encapsulating most of the wxCLIPS functionality for which it is sensible to do so. At present, it is not quite complete. wxCOOL resides in the wxcool subdirectory of the wxCLIPS installation directory, and is loaded by batching the file wx.clp. Before using the classes, you need to call the function wxcool-init.

Because wxCOOL is implemented in terms of the wxCLIPS functions, it does add overhead to a CLIPS application in terms of loading time, execution speed, and (to a less significant degree) memory requirements. So you may still wish to code speed-critical parts of your application using the raw wxCLIPS functions, especially where a lot of GUI elements are to be created.

Saving your application as a binary file will certainly speed up loading time (and help protect your source code from prying eyes) but there may be a size limit on binary files under MS Windows (as yet undetermined). Another problem is that you cannot load a binary file and then load non-binary constructs: it's all-or-nothing.

### 9.8.2. How to use the wxCOOL class reference

In the *message handler definitions* (page 18), bold words are types, and are not part of CLIPS syntax. Parameter names are in italics. Types are as follows:

- **double** is a double-precision floating point number.
- **long** is a long integer.

- **string** is a double-quoted ASCII string.
- **word** is an unquoted string.
- **bool** is a CLIPS symbol taking values TRUE or FALSE.
- **multifield** is a CLIPS multi-field value list.
- **void** means that no value is returned.

Parameters can be **optional**, in which case defaults are assumed.

Some parameters can be combinations ('bit lists') of flags. wxCLIPS mimics the compact C++ syntax by parsing strings, for example:

```
(make-instance (gensym*) of wxFrame (style "wxSDI | wxDEFAULT")
...)
```

Each identifier in such a parameter is translated to an integer value, and all are logical-or'ed together to produce an integer which is passed to the appropriate wxWindows C++ function.

**Slots** are listed before the message handlers.

**Accessors** (put-... and get-... functions) are not documented explicitly, but can be assumed where appropriate: see the documentation for each class's slots.

**Create** handlers for each class are documented, but are not explicitly called by the programmer: they are called by the *init* handler on instance creation. You can use call-next-handler from within a Create handler, to ensure all ancestors get a chance to initialise. However, you should not invoke call-next-handler in a 'delete' handler since CLIPS calls this for each class anyway.

### 9.8.3. Instance creation

Instance creation is done in the conventional CLIPS way, e.g.

```
(make-instance test1 of MyFrame (title "Hello world!") (x 20) (y 20)
(width 200) (height 200))
```

Slot initializers take the place of function parameters, which makes for more legible code, albeit for instance creation only, and not normal message passing.

Each class has a message handler called 'create', which constructs the underlying object and adds callbacks for the instance. The initialization code cannot be put into the standard 'init' handler for each class, since this is called for every class that an instance inherits from, and would result in multiple wxCLIPS objects being created for one instance. Instead, there is one init handler for wxObject which sends a create message to the object, and create is redefined for each class.

Note that the integer identifier of the underlying wxCLIPS object can be retrieved with the get-id accessor.

### 9.8.4. Types

wxCLIPS uses integers for various purposes, including boolean values. This is inconvenient in CLIPS applications, which normally use the symbolic values TRUE and FALSE. Accordingly, all wxCOOL boolean values are now symbolic (TRUE or FALSE). wxCOOL will not work correctly if you attempt to use integers instead of symbolic values.

### 9.8.5. wxCOOL event handling

wxCOOL function callbacks and events work differently from wxCLIPS callbacks and events. Instead of adding callbacks for events such as on-menu-command, you override the default event handler. All window objects derive from the class wxEvtHandler, which contains default handlers for all window callbacks. A window will normally process its own messages, so you would for example add an on-menu-command handler to your wxFrame-derived class. However, you can use the put-event-handler message to set the event handler to be a different instance of wxEvtHandler. So, you could avoid deriving from a window class altogether, and have one class which accepts the events from a variety of windows.

Panel items no longer require callback functions to be specified on creation. Instead, panel item events are sent as an on-command message to the panel item. The default wxItem on-command handler sends the message to its parent wxPanel, so you could derive a new class from wxPanel to receive on-command events, or set the item's event handler to direct it to a different instance.

The on-command handler takes wxItem instance and wxCommandEvent instance parameters. A convenient way of distinguishing incoming events is to give an item a name on creation, and test for that name in the on-command handler, using the get-name accessor.

### 9.8.6. Implementation details

From version 1.42, wxCLIPS has a few built-in functions to aid in maintaining a parallel set of classes corresponding to the underlying wxCLIPS classes (and ultimately, C++ classes). The *instance table* (page 140) functions help map between integer identifiers and CLIPS instance names. When an instance is constructed, the underlying wxCLIPS object is created and this id added to the instance table. On deletion, the entry is removed from the instance table. Callbacks are defined, such as gui-window-on-close, that are used for all instances of a class (and derived classes); they use instance-table-get-instance to retrieve the instance corresponding to the wxCLIPS object.

wxCLIPS sends OnDelete callbacks to the application when a wxCLIPS object is being deleted. This is exploited in wxCOOL to ensure that wxCOOL instances are cleaned up if wxCLIPS, and not the CLIPS application, deletes wxCLIPS objects. The twist is that we need to distinguish between an application-initiated deletion, for which we wish to call a function such as window-delete, and a wxCLIPS-initiated deletion, for which window-delete is effectively being called implicitly. To avoid deleting objects twice, wxCOOL sets a pending-delete slot in the object which is tested before deleting the underlying object.

In some cases, *all* deletions of a class's objects are initiated by wxCLIPS: for example, wxMenu instances will be deleted by the parent wxMenuBar, which is deleted implicitly when the wxFrame is deleted.

## 9.9. Resource overview

From version 1.49, wxCLIPS can load panels and dialog boxes from wxWindows resource files (extension `.wxr`). You may create dialog resources using the wxWindows Dialog Editor, which can be downloaded from:

`ftp.aiai.ed.ac.uk/pub/packages/wxwin/binary/dialoged10.zip`

Before creating a panel or dialog, load the resource file using *load-resource-file* (page 185). Then

use *panel-create-from-resource* (page 154) or *dialog-box-create-from-resource* (page 122). Alternatively you can use the wxCOOL panel or dialog box instance creation syntax, supplying the *resource* slot value).

To find an arbitrary panel item, you may need to use *find-window-by-name* (page 182) or *find-window-by-label* (page 182).

## 10. DDE commands that wxCLIPS recognizes

Under Windows, wxCLIPS functions as a simple DDE server, responding to server and topic 'WXCLIPS'. This is mainly to allow the possibility of using an external editor to load files into wxCLIPS and execute commands, offering some of the benefits of an Integrated Development Environment (IDE). It could also be used to control wxCLIPS applications remotely without having to set up a DDE server explicitly using CLIPS.

The DDE client must connect to the server using the service name WXCLIPS, and establish a conversation on the WXCLIPS topic. An example of a program that can do this is wxclipsx.exe, the source of which is provided with wxCLIPS: an executable is available from the AIAI ftp site. This may be run with the switch -c and then the DDE command to execute; if run with command line parameters, the command is executed and wxclipsx.exe exits immediately. If run without parameters, the program runs interactively and the user can type in commands. Being a wxWindows program, wxclipsx.exe is large (500K); so there is a native Windows equivalent called ddesend.exe, which is only 10 K. Ddesend takes only the DDE command, and no other switches (e.g. no -c).

The file examples/macros.rc is a sample MicroEMACS for Windows macro file, that implements commands for loading files into wxCLIPS, and executing CLIPS commands, from within MicroEMACS, using ddesend.exe.

Here are some examples of wxCLIPS DDE commands. They consist of a letter, a space, and then some data.

```
L c:\example.clp
B c:\example.clp
E (app-on-init)
Q
C
```

Here is a list of commands that wxCLIPS recognizes:

- L: load the given file of constructs into CLIPS
- B: batch the given file of constructs into CLIPS
- E: execute the given command
- C: clear the wxCLIPS development window
- Q: quit wxCLIPS



## **11. Change log**

### **11.1. Version 1.63**

August 4th, 1997

- If you supply a CLIPS file as a single argument to wxCLIPS, it will batch the file, change to that directory, and call app-on-init. This way you can have a wxCLIPS association for .clp and simply run the file.
- Better installation program provided, which registers the clp extension.

### **11.2. Version 1.62**

January 28th, 1997

- Added facename argument to font-create.

### **11.3. Version 1.61**

December 20th, 1996

- Compiled with CLIPS 6.04 (or Fuzzy CLIPS 6.04), using the latest patches taken from the CLIPS ftp site on 20th December 1996. It is now much easier to make a wxCLIPS-enabled CLIPS library - only two files need to be changed from the original CLIPS or Fuzzy CLIPS distribution.
- Compiled with wxWindows 1.66E (unreleased at this time).
- Added up-to-date Fuzzy CLIPS examples to examples/fuzzy directory.

### **11.4. Version 1.60**

August 1st, 1996

- Added an argument to list-box-set-selection to allow for deselection in a multiple-selection listbox.

### **11.5. Version 1.59**

July 11th, 1996

- Added OnAcceptConnectionEx server object callback to make implementation of wxCOOL classes easier.
- Added object-delete.

### **11.6. Version 1.58**

May 22nd, 1996

- Added fact-string-existp.

### **11.7. Version 1.57**

May 22nd, 1996

- Added frame-is-iconized, menu-bar-create-from-resource, list-box-is-selected.

- Added wxCOOL functions wxWindow::find-window-by-name, wxWindow::find-window-by-name.
- Can create wxMenuBar from resource now; see resource example.

### 11.8. Version 1.56

May 2nd, 1996

- Added window-close, window-refresh, window-set-size-hints.
- Cured some grid bugs.
- Cured a problem with reading in panels from .wxr resource files.

### 11.9. Version 1.55

April 10th, 1996

- Cured a popup-menu bug introduced in 1.55.
- Added find-window-by-name, find-window-by-label.
- Cured dc-get-text-extent-width, dc-get-text-extent-height bug.
- Added grid-get-cell-value, corrected grid-get-cols documentation.
- Added grid-on-paint, grid-on-size functions.
- Added window/object-remove-callback.

### 11.10. Version 1.55

March 26th, 1996

- Removed a Windows-resource-eating bug, showed especially when opening/closing dialogs or panels with listboxes or choice items.
- Added mkdir, rmdir, copy-file.
- Added a flags argument to file-selector to enable specification of a Save button instead of Open, for example.
- wxCLIPS save operations display Save instead of Open button.
- Added dc-set-background-mode, documented dc-set-background.

### 11.11. Version 1.54

March 10th, 1996

- Increased stack size for Windows 3.1 version.
- Fixed problem with type system that broke the Grid and HTML windows.

### 11.12. Version 1.53

March 4th, 1996

- Added OnCommand callback for panels and dialog boxes (needed when loading resources).
- Added object-get-type.
- Added window-get-next-child.
- Added resource example.
- Corrected and extended wxCLIPS/wxCOOL to handle dialog resource loading properly: see resource example.

### **11.13. Version 1.52**

February 21st, 1996

- Cured problem introduced in previous version when MDI child windows are maximized.
- Bitmaps can now be displayed in grid cells (grid-set-cell-bitmap).
- WIN32 version of make-metafile-placeable debugged.
- First Windows 95 version, with Win95 toolbars: see documentation for toolbar API changes.
- Documented OnDefaultAction panel callback for listbox double-click notification.
- Minor bug fixes to wxCOOL.

### **11.14. Version 1.51**

January 29th, 1996

- Added HTML viewer class (Windows only).

### **11.15. Version 1.50**

January 16th, 1996

- Added various functions related to canvas scrolling, plus OnScroll event.
- Windows dialog and panel default font reduced in size.
- Added grid functionality (Windows only).
- Cured bug in toolbar-add-tool where second bitmap was not recognised.
- Added get-active-window (Windows only).

### **11.16. Version 1.49**

December 21st, 1995

- Cured bug in font-create where style and weight parameters were confused.
- Added load-resource-file, clear-resources, panel-create-from-resource, dialog-box-create-from-resource.

### **11.17. Version 1.48**

November 21st, 1995

- Added more text window, multitext commands (position manipulation).
- Added app-set/get-show-frame-on-init.
- Added hwnd-send-message.

### **11.18. Version 1.47**

October, 1995.

- Added text-window-set-editable, text-window-get-contents.

### **11.19. Version 1.46**

October 16th, 1995.

- Debugged OnChar callbacks, but wxTextWindow::OnChar still has problems under Windows.
- Toolbars can now have panel items placed on them under Windows (see toolbar demo).

### **11.20. Version 1.45**

September 5th, 1995.

- Added OnCharHook for app, frame, dialog box (Windows only).
- Added app-create, key-event-... functions.

### **11.21. Version 1.44**

September, 1995.

- Added recordset-get-primary-keys, recordset-get-foreign-keys.
- Added preliminary Windows 95 support (i.e. runs under Windows 95 when compiled for Win32).
- Added get-os-platform.

### **11.22. Version 1.43**

August, 1995.

- WIN32s version introduced. Doesn't seem to work under NT or Windows 95 yet, but under WIN32s all seems well. The WIN32s version is small and faster, and blood, bsave work properly for files over 64K.
- Uses an improved version of wxWindows which tries to optimize GDI objects under MS Windows, resulting in reduced consumption of resources.
- Cured behaviour where after a CLIPS error, nothing would subsequently work (until a load, for example).

### **11.23. Version 1.42**

July, 1995.

- Added 'name' parameter to all window creation calls.
- Added window-get-name, frame-on-size.
- wxCLIPS now calls OnDelete callback when each object is being deleted.
- Started wxCOOL: an object-oriented wrapper around wxCLIPS.

### **11.24. Version 1.41**

July 12th, 1995.

- Changed radiobox semantics slightly.
- Added date class.

### **11.25. Version 1.40**

June 17th, 1995.

- Added ODBC subset support and example (Windows only).
- Started to add overviews, separate from the alphabetical reference.
- Added timer documentation.
- Added dc-get-text-extent-width, dc-get-text-extent-height.
- Added radiobox, groupbox.
- Added window-popup-menu and extra argument to menu-create for creating popup menus.
- Added command-event functions and panel-item-get-command-event for more detailed event inspection from panel item callbacks.
- Added functions: start-timer, get-elapsed-time, now, mci-send-string, bell, show-ide-window.
- Added -dir command line switch.

### 11.26. Version 1.38

April 28th, 1995.

- Cured various bugs in DDE server implementation and this manual. Sorry...
- Added ddeserv.clp, ddeclien.clp examples.

### 11.27. Version 1.36

March 14th, 1995.

- Rewrite of much of wxCLIPS: generic extension-support library (wxExtend) created for code-sharing with projects such as wxPython. Please beware of bugs introduced in this release: hopefully none, but you never know.
- Addition of *execute* function, with synchronous or asynchronous operation. Please use instead of *system*.
- DDE changes: use add-event-handler instead of the previous callback mechanism. Client DDE code has minimal change, just omit any arguments to client-create. For server operation, register interest for events OnAdvise, OnExecute, OnRequest etc., one callback function each.

### 11.28. Version 1.35

February 26th, 1995.

- Fixed bug in wxWindows where the editable text window didn't respond properly to some characters, under MS Windows.
- Added CLIPS toolbar creation functions (see examples/toolbar for a Windows demo).
- Added dc-set-background.
- Added get-ide-window.
- Added cursor-create, cursor-delete, window-set-cursor.
- message-box now has a title parameter.
- Added DDE capability under Windows, so it is possible to load files into wxCLIPS from within an editor.

### 11.29. Version 1.34

February 12th, 1995.

- Command entry panel now recognises Enter key under Windows (at last).
- Added toolbar to the development window (Windows only).

- Main text window (under Windows) is now a standard EDIT control, with better scrolling and copy to clipboard functionality.
- Added WinHelp manual logo and new Windows icon.

### **11.30. Version 1.33**

December 11th, 1994.

- Added windows printer and metafile device context support.
- Added metafile support, make-metafile-placeable.
- Added Copy, Cut, Paste to text window.
- Added begin-busy-cursor, end-busy-cursor.
- Added better file loading support to bitmap-load-from-file.
- Added basic colourmap support (bitmap-get-colourmap, dc-set-colourmap).
- Added get-resource, write-resource.

### **11.31. Version 1.32**

November 21st, 1994.

- Added PostScript device context support.
- Replaced memory-dc-delete with the more general dc-delete.
- Added timer and 'work procedure' functions.
- Added some more device context functions, e.g. dc-get-width.
- Added the file wxcitms.cc.
- Moved Clear screen menu item to File menu.

### **11.32. Version 1.30**

August 21st, 1994.

- Added bitmap and icon support (including bitmap buttons).
- Added memory device context support.
- Divided the source up a bit.

### **11.33. Versions 1.00 to 1.20**

- First and subsequent releases.

## References

*CLIPS 6.0 User Guide*, NASA Software Technology Branch

*CLIPS 6.0 Reference Manual*, NASA Software Technology Branch

*wxWindows User Manual*, Julian Smart, Artificial Intelligence Applications Institute, University of Edinburgh, 1996

## Glossary

### API

Application Programmer's Interface - a set of calls and classes defining how a library can be used.

### Bit list

A bit list in wxCLIPS is a way of specifying several window styles. It derives from C and C++ syntax, where by defining identifiers with carefully chosen binary numbers, it is possible to combine several values in one integer. In wxCLIPS, you use similar syntax to C, but enclose the list in quotes:

```
"wxCAPTION | wxMINIMIZE_BOX | wxMINIMIZE_BOX | wxTHICK_FRAME"
```

### Callback

Callbacks are application-defined functions which receive events from the GUI. You normally add a callback for a particular window (such as a canvas) and event (such as OnPaint) using window-add-callback, or pass the callback in a panel item creation function, such as button-create.

### Canvas

A canvas is a subwindow on which graphics (but not panel items) can be drawn. It may be scrollable. A canvas has a *device context overview* (page 115) associated with it.

### DDE

Dynamic Data Exchange - Microsoft's interprocess communication protocol. wxCLIPS provides a subset of DDE under both Windows and UNIX.

### Device context

A device context is an abstraction away from devices such as windows, printers and files. Code that draws to a device context is generic since that device context could be associated with a number of different real device. A canvas has a device context, although duplicate graphics calls are provided for the canvas, so the beginner doesn't have to think in terms of device contexts when starting out. See *device context overview* (page 115).

### Dialog box

In wxCLIPS a dialog box is a convenient way of popping up a window with panel items, without having to explicitly create a frame and a panel. A dialog box may be modal or modeless. A modal dialog does not return control back to the calling program until the user has dismissed it, and all other windows in the application are disabled until the dialog is dismissed. A modeless dialog is just like a normal window in that the user can access other windows while the dialog is displayed.

### Frame

A visible window usually consists of a frame which contains zero or more subwindows, such as text subwindow, canvas, and panel.

### GUI



Graphical User Interface, such as MS Windows or Motif.

### **Menu bar**

A menu bar is a series of labelled menus, usually placed near the top of a window.

### **Metafile**

MS Windows-specific object which may contain a restricted set of GDI primitives. It is device independent, since it may be scaled without losing precision, unlike a bitmap. A metafile may exist in a file or in memory. wxCLIPS implements enough metafile functionality to use it to pass graphics to other applications via the clipboard or files.

### **Open Look**

A specification for a GUI 'look and feel', initiated by Sun Microsystems. XView is one toolkit for writing Open Look applications under X, and wxCLIPS sits on top of XView (among other toolkits).

### **Panel**

A panel is a subwindow on which a limited range of panel items (widgets or controls for user input) can be placed. wxCLIPS allows panel items to be placed explicitly, or laid out from left to right, top to bottom, which is a more platform independent method since spacing is calculated automatically at run time. Panel items cannot be placed on a canvas, which is specifically for drawing graphics. However, you can draw on a panel.

### **Resource**

Resource takes several meanings in wxCLIPS. The functions `get-resource`, `write-resource` deal with MS Windows `.ini` and X `.xdefaults` resource entries. The wxWindows/wxCLIPS 'resource system', on the other hand, is a facility for loading dialog specifications from `.wxrc` files (which may be created by hand or using the wxWindows Dialog Editor).

### **Status line**

A status line is often found at the base of a window, to keep the user informed (for instance, giving a line of description to menu items, as in the **hello** demo).

### **XView**

An X toolkit supplied by Sun Microsystems for implementing the Open Look 'look and feel'. Freely available, but virtually obsolete.



## Index

### —:—

- ::ClipsErrorFunction, 7
- ::RouteCommand, 7
- ::wxCleanWindows, 7
- ::wxExecuteClipsFile, 7
- ::wxInitClips, 7
- ::wxRouteNoEcho, 8
- ::wxUserFunctions, 8

### —A—

- A selection of SQL commands, 209
- About AIAI, 2
- Accessing CLIPS C functions from C++, 8
- add-days, 37
- add-event-handlers, 66
- add-months, 33
- add-self, 38
- add-separator, 85
- add-tool, 85
- add-weeks, 33
- add-years, 33
- advise, 27
- aligning items, 197
- alt-down, 55
- app-create, 93
- append, 24, 56, 59, 60
- append-separator, 59
- app-get-show-frame-on-init, 93
- app-on-init, 93
- app-set-show-frame-on-init, 93

### —B—

- batch, 179
- begin-busy-cursor, 180
- begin-drawing, 39
- bell, 180
- bit list, 196
- bitmap, 20, 61
- bitmap-create, 94
- bitmap-delete, 94
- bitmap-get-colourmap, 94
- bitmap-get-height, 94
- bitmap-get-width, 94
- bitmap-load-from-file, 94
- bitmap-type, 18
- blit, 39
- break, 59
- brush-create, 95
- brush-delete, 95
- button, 63
- button-create, 95
- button-create-from-bitmap, 96
- button-down, 63

### —C—

- callback, 59
- canvas-create, 21, 96
- canvas-get-dc, 97
- canvas-get-scroll-page-x, 97
- canvas-get-scroll-page-y, 97
- canvas-get-scroll-pixels-per-unit-x, 98
- canvas-get-scroll-pixels-per-unit-y, 98
- canvas-get-scroll-pos-x, 97
- canvas-get-scroll-pos-y, 97
- canvas-get-scroll-range-x, 97
- canvas-get-scroll-range-y, 97
- canvas-on-char, 98
- canvas-on-scroll, 98
- canvas-scroll, 99
- canvas-set-scrollbars, 98
- canvas-set-scroll-page-x, 98
- canvas-set-scroll-page-y, 98, 99
- canvas-set-scroll-pos-x, 99
- canvas-set-scroll-pos-y, 99
- canvas-set-scroll-range-x, 99
- canvas-set-scroll-range-y, 99
- canvas-view-start-x, 99
- canvas-view-start-y, 99, 100
- centre, 89
- chdir, 180
- check, 60
- check-box-create, 100
- check-box-get-value, 100
- check-box-set-value, 100
- checked, 61
- choice-append, 101
- choice-clear, 101
- choice-create, 100
- choice-find-string, 101
- choice-get-selection, 101
- choice-get-string, 102
- choice-get-string-selection, 101
- choice-number, 102
- choice-set-selection, 101
- choice-set-string-selection, 102
- clean-windows, 180
- clear, 24, 56, 82
- clear-ide-window, 180
- clear-resources, 180
- clear-tools, 85
- client-create, 102
- client-height, 89
- client-make-connection, 102
- client-width, 89
- ClipsErrorFunction, 7
- close, 31, 63
- Code modifications, 9
- colour, 20, 69
- colour-blue, 103
- colour-create, 103

- colour-green, 103
- colour-red, 103
- command-event-get-selection, 104
- command-event-is-selection, 104
- Connection overview, 200
- connection-advise, 104
- connection-create, 104
- connection-disconnect, 105
- connection-execute, 104
- connection-poke, 105
- connection-request, 105
- connection-start-advise, 105
- connection-stop-advise, 106
- control-down, 55, 63
- copy, 82
- copy-file, 180, 181
- create, 19, 20, 21, 23, 25, 30, 33, 45, 48, 51, 52, 53, 54, 56, 58, 59, 60, 61, 62, 65, 66, 67, 69, 70, 71, 73, 80, 81, 82, 84, 86
- Create, 209
- create-buttons, 85
- create-julian, 34
- create-status-line, 49
- create-tools, 86
- cursor-create, 106
- cursor-delete, 107
- cursor-load-from-file, 107
- cursor-name, 29
- cut, 82

## —D—

- Data transfer, 200
- database, 72
- Database overview, 208
- database-close, 107
- database-create, 31, 107
- database-delete, 108
- database-error-occurred, 108
- database-get-database-name, 108
- database-get-data-source, 108
- database-get-error-code, 108
- database-get-error-message, 108
- database-get-error-number, 109
- database-is-open, 109
- database-open, 109
- date-add-days, 114
- date-add-months, 109
- date-add-self, 114
- date-add-weeks, 109
- date-add-years, 109
- date-create, 109, 110
- date-create-julian, 110
- date-create-string, 110
- date-delete, 110
- date-eq, 115
- date-format, 110
- date-ge, 115
- date-geq, 115
- date-get-day, 110
- date-get-day-of-week, 110, 111
- date-get-day-of-week-name, 111

- date-get-day-of-year, 111
- date-get-days-in-month, 111
- date-get-first-day-of-month, 111
- date-get-julian-date, 111
- date-get-month, 111
- date-get-month-end, 111
- date-get-month-name, 112
- date-get-month-start, 112
- date-get-week-of-month, 112
- date-get-week-of-year, 112
- date-get-year, 112
- date-get-year-end, 112
- date-get-year-start, 112
- date-is-leap-year, 113
- date-le, 114
- date-leq, 114
- date-neq, 115
- date-set-current-date, 113
- date-set-date, 113
- date-set-format, 113
- date-set-julian, 113
- date-set-option, 113
- date-subtract, 114
- date-subtract-days, 114
- date-subtract-self, 114
- day-of-week-name, 34
- dc, 21
- dc-begin-drawing, 115
- dc-blit, 115
- dc-clear, 116
- dc-delete, 116
- dc-destroy-clipping-region, 116
- dc-draw-ellipse, 116, 117
- dc-draw-line, 117
- dc-draw-lines, 117
- dc-draw-point, 40, 117
- dc-draw-polygon, 117
- dc-draw-rectangle, 117
- dc-draw-rounded-rectangle, 117
- dc-draw-spline, 118
- dc-draw-text, 41, 118
- dc-end-doc, 118
- dc-end-drawing, 118
- dc-end-page, 118
- dc-get-max-x, 119
- dc-get-max-y, 119
- dc-get-min-x, 118
- dc-get-min-y, 118
- dc-get-text-extent-height, 119
- dc-get-text-extent-width, 119
- dc-ok, 119
- dc-set-background, 120
- dc-set-background-mode, 120
- dc-set-brush, 120
- dc-set-clipping-region, 120
- dc-set-colourmap, 120
- dc-set-font, 120
- dc-set-logical-function, 120
- dc-set-pen, 121
- dc-set-text-background, 121
- dc-set-text-foreground, 121
- dc-start-doc, 119

dc-start-page, 119  
 debug-msg, 181  
 delete, 31, 73  
 delete-item, 57  
 depth, 19  
 destroy-clipping-region, 40  
 device, 70  
 dialog-box-create, 121  
 dialog-box-create-from-resource, 122  
 dialog-box-is-modal, 122  
 dialog-box-set-modal, 122  
 Differences in toolbar types, 204  
 dir-exists, 181  
 discard-edits, 83  
 disconnect, 27  
 display-block, 51  
 display-contents, 51  
 display-section, 52  
 dont-create, 65  
 dragging, 63  
 draw-ellipse, 40  
 draw-line, 40  
 draw-lines, 40  
 draw-polygon, 41  
 draw-rectangle, 41  
 draw-rounded-rectangle, 41  
 draw-spline, 41  
 driver, 70

## —E—

enable, 60, 61, 89  
 enable-tool, 86  
 end-busy-cursor, 181  
 end-doc, 41  
 end-drawing, 42  
 end-page, 42  
 eq, 38  
 error-occurred, 31  
 event-get-event-type, 122  
 event-position-y, 55  
 Example, 61, 147, 212  
 Examples, 201, 206  
 execute, 181  
 execute-sql, 73

## —F—

fact-string-existp, 181  
 family, 47  
 file-exists, 182  
 filename, 19, 62, 70  
 file-selector, 182  
 find-string, 24, 56  
 find-window-by-label, 90, 182  
 find-window-by-name, 89, 182  
 fit, 90  
 float-to-string, 182, 183  
 font-create, 123  
 font-delete, 123  
 format, 34  
 frame-create, 124

frame-create-status-line, 124  
 frame-iconize, 125  
 frame-is-iconized, 125  
 frame-on-size, 125  
 frame-set-icon, 125  
 frame-set-menu-bar, 125  
 frame-set-status-text, 125  
 frame-set-title, 126  
 frame-set-tool-bar, 125

## —G—

gauge-create, 128  
 gauge-set-bezel-face, 129  
 gauge-set-shadow-width, 129  
 gauge-set-value, 128  
 ge, 38  
 geq, 38  
 get-active-window, 183  
 get-char-data, 73  
 get-choice, 183  
 get-col-name, 74  
 get-col-type, 74  
 get-columns, 74  
 get-contents, 83  
 get-database-name, 32  
 get-data-source, 32  
 get-data-sources, 75  
 get-day, 34  
 get-day-of-week, 34  
 get-day-of-week-name, 111  
 get-day-of-year, 34  
 get-days-in-month, 35  
 get-elapsed-time, 183  
 get-error-code, 75  
 get-error-message, 32  
 get-error-number, 32  
 get-event-type, 47  
 get-filter, 75  
 get-first-day-of-month, 35  
 get-first-selection, 58  
 get-float-data, 75  
 get-foreign-keys, 75  
 get-ide-window, 183  
 get-int-data, 76  
 get-julian-date, 35  
 get-key-code, 55  
 get-label, 68  
 get-max-height, 86  
 get-max-width, 87  
 get-max-x, 42  
 get-max-y, 42  
 get-min-x, 42  
 get-min-y, 42  
 get-month, 35  
 get-month-end, 35  
 get-month-name, 35  
 get-month-start, 35  
 get-name, 90  
 get-next-selection, 58  
 get-number-cols, 76  
 get-number-fields, 76

get-number-params, 76  
 get-number-records, 77  
 get-os-version, 183  
 get-parent, 90  
 get-platform, 184  
 get-primary-keys, 77  
 get-resource, 184  
 get-result-set, 77  
 get-selection, 24, 26, 57, 72  
 get-string, 25, 57  
 get-string-selection, 24, 57  
 get-table-name, 77  
 get-tables, 77  
 get-text-extent-height, 42  
 get-text-extent-width, 42  
 get-text-from-user, 184  
 get-tool-client-data, 87  
 get-tool-enabled, 87  
 get-tool-long-help, 87  
 get-tool-short-help, 87  
 get-tool-state, 87  
 get-week-of-month, 35  
 get-week-of-year, 36  
 get-year, 36  
 get-year-end, 36  
 get-year-start, 36  
 goto, 78  
 grid-adjust-scrollbar, 129  
 grid-append-cols, 129  
 grid-append-rows, 129  
 grid-clear-grid, 130  
 grid-create, 130  
 grid-create-grid, 130  
 grid-delete-cols, 130  
 grid-delete-rows, 130  
 grid-get-cell-alignment, 130  
 grid-get-cell-background-colour, 130  
 grid-get-cell-bitmap, 131  
 grid-get-cell-text-colour, 131  
 grid-get-cell-value, 131  
 grid-get-cols, 132  
 grid-get-column-width, 131  
 grid-get-cursor-column, 131  
 grid-get-cursor-row, 131  
 grid-get-editable, 132  
 grid-get-label-alignment, 132  
 grid-get-label-background-colour, 132  
 grid-get-label-size, 132  
 grid-get-label-text-colour, 132  
 grid-get-label-value, 132  
 grid-get-row-height, 133  
 grid-get-rows, 131  
 grid-get-scroll-pos-x, 133  
 grid-get-scroll-pos-y, 133  
 grid-get-text-item, 133  
 grid-insert-cols, 133  
 grid-insert-rows, 133  
 grid-on-activate, 133  
 grid-on-paint, 134  
 grid-on-size, 134  
 grid-set-cell-alignment, 134  
 grid-set-cell-background-colour, 131, 134

grid-set-cell-bitmap, 134  
 grid-set-cell-text-colour, 131, 134  
 grid-set-cell-text-font, 134, 135  
 grid-set-cell-value, 135  
 grid-set-column-width, 135  
 grid-set-divider-pen, 135  
 grid-set-editable, 135  
 grid-set-grid-cursor, 135  
 grid-set-label-alignment, 135  
 grid-set-label-background-colour, 135, 136  
 grid-set-label-size, 136  
 grid-set-label-text-colour, 136  
 grid-set-label-text-font, 136  
 grid-set-label-value, 136  
 grid-set-row-height, 136  
 grid-update-dimensions, 136  
 group-box-create, 137

## —H—

height, 19, 54, 89  
 help-create, 126  
 help-delete, 126  
 help-display-block, 126  
 help-display-contents, 126  
 help-display-section, 126  
 help-keyword-search, 127  
 help-load-file, 127  
 How to use the wxCOOL class reference, 213  
 html-back, 137  
 html-cancel, 137  
 html-clear-cache, 137  
 html-create, 138  
 html-get-current-url, 138  
 html-on-size, 138  
 html-open-file, 138  
 html-open-url, 138  
 html-resize, 138  
 html-save-file, 138  
 hwnd-find, 127  
 hwnd-iconize, 127  
 hwnd-move, 127  
 hwnd-quit, 128  
 hwnd-refresh, 127  
 hwnd-send-message, 127, 128  
 hwnd-show, 128

## —I—

icon-create, 139  
 icon-delete, 139  
 icon-get-height, 139  
 icon-get-width, 139  
 iconize, 49  
 icon-load-from-file, 139  
 id, 65  
 Implementation details, 215  
 init after, 66  
 Insert, 209  
 Instance creation, 214  
 instance-table-add-entry, 140  
 instance-table-delete-entry, 140

instance-table-get-instance, 140  
 interactive, 70, 71  
 is-bof, 78  
 is-button, 64  
 is-col-nullable, 78  
 is-eof, 78  
 is-field-dirty, 78  
 is-field-null, 78  
 is-leap-year, 36  
 is-open, 32, 78  
 is-selection, 26

—K—

key-event-alt-down, 140  
 key-event-control-down, 140  
 key-event-get-key-code, 141  
 key-event-position-x, 141  
 key-event-position-y, 141  
 key-event-shift-down, 141  
 keyword-search, 52

—L—

layout, 87  
 le, 38  
 left-down, 63  
 left-up, 63  
 leq, 38  
 list-box-append, 142  
 list-box-clear, 142  
 list-box-create, 141  
 list-box-delete, 143  
 list-box-find-string, 142  
 list-box-get-first-selection, 143  
 list-box-get-next-selection, 143  
 list-box-get-selection, 142  
 list-box-get-string, 143  
 list-box-get-string-selection, 142  
 list-box-is-selected, 142  
 list-box-number, 143  
 list-box-set-selection, 142  
 list-box-set-string-selection, 143  
 load-file, 52, 83  
 load-resource-file, 185  
 long-to-string, 185

—M—

major-dimension, 71  
 make-connection, 25  
 make-metafile-placeable, 185  
 make-modal, 90  
 max, 80  
 mci-send-string, 185  
 memory-dc-create, 143  
 memory-dc-select-object, 144  
 menu-append, 144, 145  
 menu-append-separator, 145  
 menu-bar-append, 146  
 menu-bar-check, 146  
 menu-bar-checked, 146

menu-bar-create, 145, 146  
 menu-bar-create-from-resource, 146  
 menu-bar-enable, 146  
 menu-break, 145  
 menu-check, 145  
 menu-create, 144  
 menu-enable, 145  
 message-box, 186  
 message-create, 146  
 message-create-from-bitmap, 147  
 metafile-dc-close, 148  
 metafile-dc-create, 148  
 metafile-delete, 147  
 metafile-set-clipboard, 148  
 middle-down, 64  
 middle-up, 64  
 min, 80  
 mkdir, 187  
 modal, 45  
 modified, 83  
 mouse-event-button, 148  
 mouse-event-button-down, 149  
 mouse-event-control-down, 149  
 mouse-event-dragging, 149  
 mouse-event-is-button, 149  
 mouse-event-left-down, 149  
 mouse-event-left-up, 149  
 mouse-event-middle-down, 149  
 mouse-event-middle-up, 149  
 mouse-event-position-x, 150  
 mouse-event-position-y, 150  
 mouse-event-right-down, 150  
 mouse-event-right-up, 150  
 mouse-event-shift-down, 150  
 move, 78  
 move-first, 79  
 move-last, 79  
 move-next, 79  
 move-prev, 79  
 multiple, 56  
 multi-text-copy, 151  
 multi-text-create, 150  
 multi-text-cut, 151  
 multi-text-get-insertion-point, 151  
 multi-text-get-last-position, 151  
 multi-text-get-line-length, 151  
 multi-text-get-line-text, 152  
 multi-text-get-number-of-lines, 152  
 multi-text-get-value, 152  
 multi-text-paste, 152  
 multi-text-position-to-char, 152  
 multi-text-position-to-line, 152  
 multi-text-remove, 152  
 multi-text-replace, 153  
 multi-text-set-insertion-point, 153  
 multi-text-set-selection, 153  
 multi-text-set-value, 152  
 multi-text-show-position, 153  
 multi-text-write, 153  
 multi-text-xy-to-position, 153

—N—

native, 51  
 neq, 39  
 new-line, 68  
 now, 187  
 number, 57

—O—

object-delete, 153  
 object-get-type, 153, 154  
 ODBC SQL data types, 208  
 ok, 43  
 on-accept-connection, 80  
 on-activate, 50  
 on-advise, 28  
 on-char, 22  
 on-char-hook, 18, 46, 50  
 on-close, 46, 50  
 on-command, 67, 68  
 on-default-action, 67  
 on-event, 22  
 on-execute, 28  
 on-make-connection, 25  
 on-menu-command, 50  
 on-menu-select, 50  
 on-paint, 22, 46, 88  
 on-poke, 28  
 on-request, 29  
 on-size, 22, 46, 51  
 on-start-advise, 29  
 on-stop-advise, 29  
 open, 33  
 orientation, 85

—P—

panel-create, 154  
 panel-create-from-resource, 154  
 panel-item-get-command-event, 155  
 panel-item-get-label, 155, 156  
 panel-item-set-default, 156  
 panel-item-set-label, 156  
 panel-new-line, 155  
 panel-set-button-font, 155  
 panel-set-label-font, 155  
 panel-set-label-position, 155  
 paste, 83  
 pen-create, 156  
 pen-delete, 156  
 pending-delete, 66  
 point-size, 47  
 poke, 27  
 popup-menu, 90  
 position-x, 55, 64  
 position-y, 64  
 postscript-dc-create, 157  
 printer-dc-create, 157  
 Procedures for writing an ODBC application, 205

—Q—

query, 79

—R—

radio-box-create, 157  
 radio-box-get-selection, 158  
 radio-box-set-selection, 158  
 range, 52  
 read-string, 187  
 Recordset overview, 208  
 recordset-create, 158  
 recordset-delete, 158  
 recordset-execute-sql, 158  
 recordset-get-char-data, 159  
 recordset-get-col-name, 159  
 recordset-get-col-type, 159  
 recordset-get-columns, 159  
 recordset-get-database, 160  
 recordset-get-data-sources, 160  
 recordset-get-error-code, 160  
 recordset-get-filter, 160  
 recordset-get-float-data, 160  
 recordset-get-foreign-keys, 161  
 recordset-get-int-data, 161  
 recordset-get-number-cols, 161, 162  
 recordset-get-number-fields, 162  
 recordset-get-number-params, 162  
 recordset-get-number-records, 162  
 recordset-get-primary-keys, 162  
 recordset-get-result-set, 162  
 recordset-get-table-name, 162  
 recordset-get-tables, 163  
 recordset-goto, 163  
 recordset-is-bof, 163  
 recordset-is-col-nullable, 163  
 recordset-is-eof, 164  
 recordset-is-field-dirty, 163  
 recordset-is-field-null, 163  
 recordset-is-open, 164  
 recordset-move, 164  
 recordset-move-first, 164  
 recordset-move-last, 164  
 recordset-move-next, 164  
 recordset-move-prev, 164  
 recordset-query, 164  
 recordset-set-table-name, 164, 165  
 request, 27  
 resource, 67  
 Restrictions, 11  
 return-result, 187  
 right-down, 64  
 right-up, 64  
 rmdir, 187  
 RouteCommand, 7  
 rows-or-columns, 85

—S—

save-file, 83  
 scroll, 22



Select, 209  
 select-object, 58  
 server-create, 165  
 service-name, 26, 79  
 set, 36  
 set-background, 43  
 set-background-mode, 43  
 set-bezel-face, 53  
 set-brush, 43  
 set-button-font, 68  
 set-client-size, 91  
 set-clipping-region, 44  
 set-colourmap, 43  
 set-cursor, 91  
 set-date, 37  
 set-default, 69  
 set-default-size, 88  
 set-editable, 84  
 set-focus, 91  
 set-font, 44  
 set-format, 37  
 set-icon, 49  
 set-julian, 36  
 set-label, 69  
 set-label-font, 68  
 set-label-position, 68  
 set-logical-function, 44  
 set-margins, 88  
 set-menu-bar, 49  
 set-option, 37  
 set-pen, 44  
 set-scrollbar, 22  
 set-selection, 24, 57, 72  
 set-shadow-width, 53  
 set-size, 91  
 set-status-text, 49  
 set-string-selection, 25, 57  
 set-table-name, 79  
 set-text-background, 44  
 set-text-foreground, 44  
 set-title, 49  
 set-tool-bar, 50  
 set-tool-long-help, 88  
 set-tool-short-help, 88  
 set-value, 82  
 set-work-proc, 187  
 shift-down, 55, 65  
 show, 91  
 show-ide-window, 187  
 sleep, 188  
 slider-create, 165  
 slider-get-value, 166  
 slider-set-value, 166  
 start, 84  
 start-advise, 28  
 start-doc, 43  
 start-page, 43  
 start-timer, 188  
 stop, 84  
 stop-advise, 28  
 string-sort, 188  
 string-to-float, 188

string-to-long, 188  
 string-to-symbol, 188  
 style, 20, 47, 69, 196  
 subtract, 38  
 subtract-days, 37  
 subtract-self, 38  
 symbol-to-string, 189

## —T—

text-create, 166  
 text-get-value, 167  
 text-set-value, 167  
 text-window-clear, 167  
 text-window-copy, 168  
 text-window-create, 168  
 text-window-cut, 168  
 text-window-discard-edits, 168  
 text-window-get-contents, 168  
 text-window-get-insertion-point, 168  
 text-window-get-last-position, 169  
 text-window-get-line-length, 169  
 text-window-get-line-text, 169  
 text-window-get-number-of-lines, 169  
 text-window-load-file, 169  
 text-window-modified, 169  
 text-window-on-char, 169  
 text-window-paste, 169, 170  
 text-window-position-to-char, 170  
 text-window-position-to-line, 170  
 text-window-remove, 170  
 text-window-replace, 170  
 text-window-save-file, 170  
 text-window-set-editable, 170  
 text-window-set-insertion-point, 171  
 text-window-set-selection, 171  
 text-window-show-position, 170  
 text-window-write, 171  
 text-window-xy-to-position, 171  
 The appearance and behaviour of a grid, 211  
 timer-create, 171  
 timer-delete, 171  
 timer-start, 171  
 timer-stop, 172  
 toggle-tool, 88  
 toolbar-add-separator, 172  
 toolbar-add-tool, 172  
 toolbar-clear-tools, 172  
 toolbar-create, 172  
 toolbar-create-tools, 173  
 toolbar-enable-tool, 173  
 toolbar-get-max-height, 173  
 toolbar-get-max-width, 173  
 toolbar-get-tool-client-data, 173  
 toolbar-get-tool-enabled, 174  
 toolbar-get-tool-long-help, 174  
 toolbar-get-tool-short-help, 174  
 toolbar-get-tool-state, 174  
 toolbar-layout, 174  
 toolbar-on-paint, 174  
 toolbar-set-default-size, 174  
 toolbar-set-margins, 175

toolbar-set-tool-long-help, 175  
 toolbar-set-tool-short-help, 175  
 toolbar-toggle-tool, 175  
 type, 72  
 Types, 214

## —U—

underlined, 47  
 Update, 210

## —V—

value, 23, 52, 80, 81  
 values, 23, 56, 71

## —W—

weight, 47  
 What is wxCOOL?, 213  
 width, 19, 54, 89  
 window, 70, 71  
 window-add-callback, 175  
 window-centre, 175  
 window-close, 175  
 window-delete, 176  
 window-enable, 176  
 window-fit, 176  
 window-get-client-height, 177  
 window-get-client-width, 177  
 window-get-height, 177  
 window-get-name, 176  
 window-get-next-child, 176  
 window-get-parent, 177  
 window-get-width, 177  
 window-get-x, 177  
 window-get-y, 177  
 window-is-shown, 178  
 window-make-modal, 178  
 window-popup-menu, 178  
 window-refresh, 178  
 window-remove-callback, 178  
 window-set-client-size, 179  
 window-set-cursor, 178  
 window-set-focus, 179  
 window-set-size, 179  
 window-set-size-hints, 179  
 window-show, 179  
 write, 84  
 write-resource, 189  
 wxALWAYS\_SB, 197  
 wxApplication on-char-hook, 18  
 wxBitmap bitmap-type, 18  
 wxBitmap create, 19  
 wxBitmap depth, 19  
 wxBitmap filename, 19  
 wxBitmap height, 19  
 wxBitmap width, 19  
 wxBORDER, 198, 199  
 wxBrush colour, 20  
 wxBrush create, 20  
 wxBrush style, 20

wxBUTTON bitmap, 20  
 wxButton create, 20  
 wxButton styles, 197  
 wxCanvas create, 21  
 wxCanvas dc, 21  
 wxCanvas on-char, 22  
 wxCanvas on-event, 22  
 wxCanvas on-paint, 22  
 wxCanvas on-size, 22  
 wxCanvas scroll, 22  
 wxCanvas set-scrollbars, 22  
 wxCanvas styles, 199  
 wxCAPTION, 196  
 wxCheckBox create, 23  
 wxCheckBox value, 23  
 wxChoice append, 24  
 wxChoice clear, 24  
 wxChoice create, 23  
 wxChoice find-string, 24  
 wxChoice get-selection, 24  
 wxChoice get-string, 25  
 wxChoice get-string-selection, 24  
 wxChoice set-selection, 24  
 wxChoice set-string-selection, 24  
 wxChoice values, 23  
 wxCleanWindows, 7  
 wxClient create, 25  
 wxClient make-connection, 25  
 wxClient on-make-connection, 25  
 wxclips-object-exists, 189  
 wxCommandEvent get-selection, 26  
 wxCommandEvent is-selection, 26  
 wxConnection advise, 27  
 wxConnection disconnect, 27  
 wxConnection execute, 27  
 wxConnection on-advise, 28  
 wxConnection on-execute, 28  
 wxConnection on-poke, 28  
 wxConnection on-request, 29  
 wxConnection on-start-advise, 29  
 wxConnection on-stop-advise, 29  
 wxConnection poke, 27  
 wxConnection request, 27  
 wxConnection service-name, 26  
 wxConnection start-advise, 28  
 wxConnection stop-advise, 28  
 wxCOOL event handling, 215  
 wxCursor create, 30  
 wxCursor cursor-name, 29  
 wxCursor x, 30  
 wxCursor y, 30  
 wxDatabase close, 31  
 wxDatabase create, 31  
 wxDatabase delete, 31  
 wxDatabase error-occurred, 31  
 wxDatabase get-database-name, 32  
 wxDatabase get-data-source, 32  
 wxDatabase get-error-code, 32  
 wxDatabase get-error-message, 32  
 wxDatabase get-error-number, 32  
 wxDatabase is-open, 32  
 wxDatabase open, 32

- wxDc add-days, 37
- wxDc add-months, 33
- wxDc add-self, 38
- wxDc add-weeks, 33
- wxDc add-years, 33
- wxDc create, 33
- wxDc create-julian, 34
- wxDc create-string, 34
- wxDc eq, 38
- wxDc format, 34
- wxDc ge, 38
- wxDc geq, 38
- wxDc get-day, 34
- wxDc get-day-of-week, 34
- wxDc get-day-of-week-name, 34
- wxDc get-day-of-year, 34
- wxDc get-days-in-month, 34
- wxDc get-first-day-of-month, 35
- wxDc get-julian-date, 35
- wxDc get-month, 35
- wxDc get-month-end, 35
- wxDc get-month-name, 35
- wxDc get-month-start, 35
- wxDc get-week-of-month, 35
- wxDc get-week-of-year, 36
- wxDc get-year, 36
- wxDc get-year-end, 36
- wxDc get-year-start, 36
- wxDc is-leap-year, 36
- wxDc le, 38
- wxDc leq, 38
- wxDc neq, 39
- wxDc set, 36
- wxDc set-date, 36
- wxDc set-format, 37
- wxDc set-julian, 36
- wxDc set-option, 37
- wxDc subtract, 37
- wxDc subtract-days, 37
- wxDc subtract-self, 38
- wxDc begin-drawing, 39
- wxDc blit, 39
- wxDc clear, 40
- wxDc destroy-clipping-region, 40
- wxDc draw-ellipse, 40
- wxDc draw-line, 40
- wxDc draw-lines, 40
- wxDc draw-point, 40
- wxDc draw-polygon, 41
- wxDc draw-rectangle, 41
- wxDc draw-rounded-rectangle, 41
- wxDc draw-spline, 41
- wxDc draw-text, 41
- wxDc end-doc, 41
- wxDc end-drawing, 41
- wxDc end-page, 42
- wxDc get-max-x, 42
- wxDc get-max-y, 42
- wxDc get-min-x, 42
- wxDc get-min-y, 42
- wxDc get-text-extent-height, 42
- wxDc get-text-extent-width, 42
- wxDc ok, 43
- wxDc set-background, 43
- wxDc set-background-mode, 43
- wxDc set-brush, 43
- wxDc set-clipping-region, 44
- wxDc set-colourmap, 43
- wxDc set-font, 44
- wxDc set-logical-function, 44
- wxDc set-pen, 44
- wxDc set-text-background, 44
- wxDc set-text-foreground, 44
- wxDc start-doc, 43
- wxDc start-page, 43
- wxDEFAULT\_DIALOG\_STYLE, 196
- wxDEFAULT\_FRAME, 196
- wxDialogBox create, 45
- wxDialogBox modal, 45
- wxDialogBox on-char-hook, 46
- wxDialogBox on-close, 46
- wxDialogBox on-paint, 46
- wxDialogBox on-size, 46
- wxDialogBox styles, 196
- wxEvent get-event-type, 46
- wxExecuteClipsFile, 7
- wxFIXED\_LENGTH, 197
- wxFont create, 47
- wxFont family, 47
- wxFont point-size, 47
- wxFont style, 47
- wxFont underlined, 47
- wxFont weight, 47
- wxFrame create, 48
- wxFrame create-status-line, 49
- wxFrame iconize, 49
- wxFrame on-activate, 50
- wxFrame on-char-hook, 50
- wxFrame on-close, 50
- wxFrame on-menu-command, 50
- wxFrame on-menu-select, 50
- wxFrame on-size, 51
- wxFrame set-icon, 49
- wxFrame set-menu-bar, 49
- wxFrame set-status-text, 49
- wxFrame set-title, 49
- wxFrame set-tool-bar, 50
- wxFrame styles, 196
- wxGA\_HORIZONTAL, 197
- wxGA\_PROGRESSBAR, 197
- wxGA\_VERTICAL, 197
- wxGauge create, 52
- wxGauge range, 52
- wxGauge set-bezel-face, 53
- wxGauge set-shadow-width, 53
- wxGauge styles, 197
- wxGauge value, 52
- wxGroupBox create, 53
- wxGroupBox styles, 197
- wxHelpInstance create, 51
- wxHelpInstance display-block, 51
- wxHelpInstance display-contents, 51
- wxHelpInstance display-section, 52
- wxHelpInstance keyword-search, 52

- wxHelpInstance load-file, 52
- wxHelpInstance native, 51
- wxHORIZONTAL, 198
- wxHORIZONTAL\_LABEL, 197
- wxHSCROLL, 197, 198
- wxIcon create, 54
- wxIcon height, 53
- wxIcon width, 54
- wxICONIZE, 196
- wxInitClips, 8
- wxItem get-label, 68
- wxItem on-command, 68
- wxItem set-default, 69
- wxItem set-label, 69
- wxItem styles, 197
- wxKeyEvent alt-down, 55
- wxKeyEvent control-down, 55
- wxKeyEvent get-key-code, 55
- wxKeyEvent position-x, 55
- wxKeyEvent position-y, 55
- wxKeyEvent shift-down, 55
- wxLB\_ALWAYS\_SB, 197
- wxLB\_EXTENDED, 197
- wxLB\_MULTIPLE, 197
- wxLB\_NEEDED\_SB, 197
- wxLB\_SINGLE, 197
- wxListBox append, 56
- wxListBox clear, 56
- wxListBox create, 56
- wxListBox delete-item, 57
- wxListBox find-string, 56
- wxListBox get-first-selection, 58
- wxListBox get-next-selection, 58
- wxListBox get-selection, 57
- wxListBox get-string, 57
- wxListBox get-string-selection, 57
- wxListBox multiple, 56
- wxListBox number, 57
- wxListBox set-selection, 57
- wxListBox set-string-selection, 57
- wxListBox styles, 197
- wxListBox values, 55
- wxMAXIMIZE, 196
- wxMAXIMIZE\_BOX, 196
- wxMDI\_CHILD, 196
- wxMDI\_PARENT, 196
- wxMemoryDC create, 58
- wxMemoryDC select-object, 58
- wxMenu append, 59
- wxMenu append-separator, 59
- wxMenu break, 59
- wxMenu callback, 59
- wxMenu check, 60
- wxMenu create, 59
- wxMenu enable, 60
- wxMenuBar append, 60
- wxMenuBar check, 60
- wxMenuBar checked, 60
- wxMenuBar create, 60
- wxMenuBar enable, 61
- wxMessage bitmap, 61
- wxMessage create, 61
- wxMessage styles, 197
- wxMetaFile set-clipboard, 62
- wxMetaFileDC close, 63
- wxMetafileDC create, 62
- wxMetaFileDC filename, 62
- wxMINIMIZE, 196
- wxMINIMIZE\_BOX, 196
- wxMouseEvent button, 63
- wxMouseEvent button-down, 63
- wxMouseEvent control-down, 63
- wxMouseEvent dragging, 63
- wxMouseEvent is-button, 64
- wxMouseEvent left-down, 63
- wxMouseEvent left-up, 63
- wxMouseEvent middle-down, 64
- wxMouseEvent middle-up, 64
- wxMouseEvent position-x, 64
- wxMouseEvent position-y, 64
- wxMouseEvent right-down, 64
- wxMouseEvent right-up, 64
- wxMouseEvent shift-down, 64
- wxMultiText create, 65
- wxNATIVE\_IMPL, 198
- wxNEEDED\_SB, 197
- wxObject add-event-handlers, 66
- wxObject create, 66
- wxObject dont-create, 65
- wxObject id, 65
- wxObject init after, 66
- wxObject pending-delete, 66
- wxPanel create, 67
- wxPanel new-line, 68
- wxPanel on-command, 67
- wxPanel resource, 67
- wxPanel set-button-font, 68
- wxPanel set-label-font, 68
- wxPanel set-label-position, 68
- wxPanel styles, 198
- wxPen colour, 69
- wxPen create, 69
- wxPen style, 69
- wxPostScriptDC create, 70
- wxPostScriptDC filename, 70
- wxPostScriptDC interactive, 70
- wxPostScriptDC window, 70
- wxPrinter create, 71
- wxPrinterDC device, 70
- wxPrinterDC driver, 70
- wxPrinterDC filename, 70
- wxPrinterDC interactive, 71
- wxPrinterDC window, 71
- wxRadioBox, 198
- wxRadioBox create, 71
- wxRadioBox get-selection, 72
- wxRadioBox major-dimension, 71
- wxRadioBox set-selection, 72
- wxRadioBox values, 71
- wxREADONLY, 198
- wxRecordSet create, 73
- wxRecordSet database, 72
- wxRecordSet delete, 73
- wxRecordSet execute-sql, 73

wxRecordSet get-char-data, 73  
wxRecordSet get-col-name, 74  
wxRecordSet get-col-type, 74  
wxRecordSet get-columns, 74  
wxRecordSet get-data-sources, 75  
wxRecordSet get-error-code, 75  
wxRecordSet get-filter, 75  
wxRecordSet get-float-data, 75  
wxRecordSet get-foreign-keys, 75  
wxRecordSet get-int-data, 76  
wxRecordSet get-number-cols, 76  
wxRecordSet get-number-fields, 76  
wxRecordSet get-number-params, 76  
wxRecordSet get-number-records, 77  
wxRecordSet get-primary-keys, 77  
wxRecordSet get-result-set, 77  
wxRecordSet get-table-name, 77  
wxRecordSet get-tables, 77  
wxRecordSet goto, 78  
wxRecordSet is-bof, 78  
wxRecordSet is-col-nullable, 78  
wxRecordSet is-eof, 78  
wxRecordSet is-field-dirty, 78  
wxRecordSet is-field-null, 78  
wxRecordSet is-open, 78  
wxRecordSet move, 78  
wxRecordSet move-first, 79  
wxRecordSet move-last, 79  
wxRecordSet move-next, 79  
wxRecordSet move-prev, 79  
wxRecordSet query, 79  
wxRecordSet set-table-name, 79  
wxRecordSet type, 72  
wxRESIZE\_BORDER, 196  
wxRETAINED, 199  
wxRouteNoEcho, 8  
wxSDI, 196  
wxServer create, 79  
wxServer on-accept-connection, 80  
wxServer service-name, 79  
wxSlider create, 80  
wxSlider max, 80  
wxSlider min, 80  
wxSlider styles, 198  
wxSlider value, 80  
wxSTAY\_ON\_TOP, 196  
wxSYSTEM\_MENU, 196  
wxTB\_3DBUTTONS, 199  
wxTE\_PASSWORD, 198  
wxTE\_PROCESS\_ENTER, 198  
wxTE\_READONLY, 198  
wxText create, 81  
wxText set-value, 82  
wxText value, 81  
wxText/wxMultiText styles, 198  
wxTextWindow clear, 82  
wxTextWindow copy, 82  
wxTextWindow create, 82  
wxTextWindow cut, 82  
wxTextWindow discard-edits, 83  
wxTextWindow get-contents, 83  
wxTextWindow load-file, 83  
wxTextWindow modified, 83  
wxTextWindow paste, 83  
wxTextWindow save-file, 83  
wxTextWindow set-editable, 84  
wxTextWindow styles, 198  
wxTextWindow write, 84  
wxTHICK\_FRAME, 196  
wxTimer create, 84  
wxTimer start, 84  
wxTimer stop, 84  
wxTINY\_CAPTION\_HORIZ, 196  
wxTINY\_CAPTION\_VERT, 196  
wxToolBar add-separator, 85  
wxToolBar add-tool, 85  
wxToolBar clear-tools, 85  
wxToolBar create, 86  
wxToolBar create-buttons, 84  
wxToolBar create-tools, 86  
wxToolBar enable-tool, 86  
wxToolBar get-max-height, 86  
wxToolBar get-max-width, 87  
wxToolBar get-tool-client-data, 87  
wxToolBar get-tool-enabled, 87  
wxToolBar get-tool-long-help, 87  
wxToolBar get-tool-short-help, 87  
wxToolBar get-tool-state, 87  
wxToolBar layout, 87  
wxToolBar on-paint, 87  
wxToolBar orientation, 85  
wxToolBar rows-or-columns, 85  
wxToolBar set-default-size, 88  
wxToolBar set-margins, 88  
wxToolBar set-tool-long-help, 88  
wxToolBar set-tool-short-help, 88  
wxToolBar styles, 199  
wxToolBar toggle-tool, 88  
wxUSER\_COLOURS, 196, 198  
wxUserFunctions, 8  
wxVERTICAL, 198  
wxVERTICAL\_LABEL, 197  
wxVSCROLL, 196, 198  
wxWindow centre, 89  
wxWindow client-height, 89  
wxWindow client-width, 89  
wxWindow enable, 89  
wxWindow find-window-by-label, 90  
wxWindow find-window-by-name, 89  
wxWindow fit, 90  
wxWindow get-name, 90  
wxWindow get-parent, 90  
wxWindow height, 89  
wxWindow make-modal, 90  
wxWindow popup-menu, 90  
wxWindow set-client-size, 91  
wxWindow set-cursor, 90  
wxWindow set-focus, 91  
wxWindow set-size, 91  
wxWindow show, 91  
wxWindow width, 89  
wxWindow x, 88  
wxWindow y, 88

—X—

x, 30, 88

—Y—

y, 30, 89  
yield, 189