17
Resources

In the X Window System, the user can configure the interface according to options available in specific applications. The user accomplishes this through a *resource database* that resides in the X server. The X Protocol provides many ways to access the resource database, as well as many functions to aid in this task. You should consult Chapter 11, *Managing User Preferences*, in Volume One, *Xlib Programming Manual*, for a complete, in-depth discussion of X resource specification and management. Related programs include *xrdb*, and related functions can be found in Volume Two, *Xlib Reference Manual*.

XView provides many functions that allow the programmer to interact with the server to get or set resources specified by the user. Robust applications should account for user-definable defaults. That is, your programs should always consider the user's wishes for changing attribute values for things like fonts, colors and maybe even window sizes, as long as the values do not interfere with the normal running of the program. You should also provide the user with a list of resources that can be set, and you should test for them in your application.

17.1 Predefined Defaults

All of the packages in XView look for predefined defaults that the user can set in his/her resource environment. Table 17-1 outlines these defaults, their types and legal values. Refer to Section 6, *Command-line Arguments and XView Resources*, in the *XView Reference Manual* for a description of each of the XView resources. Note that the term *maxint* refers to the maximum value for an integer on your particular machine. These values tend to be rather large with respect to the intended values used by these resources. It is up to the user to use "reasonable" values.

NOTE

Table 17-1 does not include the mouseless model resources. Refer to Section 6, *Command-line Arguments and XView Resources*, in the *XView Reference Manual* for a list of the mouseless resources.

Table 17-1. Resources and Default Values Understood by XView

Name	Type	Default	Legal Values
alarm.audible	**boolean**	True	True, False
alarm.visible	**boolean**	True	True, False
cmdtool.checkpointFrequency	**integer**	0	0, <*maxint*>
cmdtool.maxLogFileSize	**integer**	<*maxint*>	0, <*maxint*>
font.name	**string**	NULL	*fontname-size*
icon.font.name	**string**	NULL	*fontname-size*
icon.pixmap	**string**	NULL	*pixmap filename*
icon.footer	**string**	NULL	*footer string*
icon.x	**integer**	0	—
icon.y	**integer**	0	—
keyboard.arrowKeys	**string**	Yes	Yes, No
keyboard.leftHanded	**string**	No	Yes, No
mouse.multiclick.space	**integer**		
notice.PopupJumpCursor	**boolean**	True	True, False
notice.beepCount	**integer**	1	—
openWindows.DragRightDistance	**integer**	100	0, <maxint>
openWindows.3DLook.color	**boolean**	True	True, False
openWindows.3DLook.monochrome	**boolean**	True	True, False
openWindows.multiClickTimeout	**integer**	4	1 through 10 (in sec/10)
openWindows.dragRightDistance	**integer**	5	—
openWindows.scrollbarPlacement	**string**	right	Left, Right
OpenWindows.MonospaceFont	**string**	NULL	fontname-size
OpenWindows.RegularFont	**string**	NULL	fontname-size
OpenWindows.BoldFont	**string**	NULL	fontname-size
OpenWindows.Scale	**string**	NULL	Small, Medium, Large, Extra-Large
scrollbar.repeatDelay	**integer**	100	0, 999 (msec delay interval)
scrollbar.pageInterval	**integer**	100	0, 999 (msec delay interval)
scrollbar.lineInterval	**integer**	1	0, 999 (msec delay interval)
server.name	**string**	getenv(DISPLAY)	*hostnamedisplay*
term.boldStyle	**string**	Invert	xview.ICCCMCompli- ant@boolean@True@True, False None, OFFSET_X, OFFSET_Y, OFFSET_X_AND_Y, OFFSET_X_AND_XY, OFFSET_Y_AND_XY, OFFSET_X_AND_Y_AND_XY, OFFSET_XY, INVERT
term.enableEdit	**boolean**	True	True, False
term.inverseStyle	**string**	Enable	ENABLED, DISABLED, SAME_AS_BOLD
term.underlineStyle	**string**	Enable	ENABLED, DISABLED, SAME_AS_BOLD
text.againLimit	**integer**	1	0, 500
text.autoIndent	**boolean**	False	True, False
text.autoScrollBy	**integer**	1	0, 100

Table 17-1. Resources and Default Values Understood by XView (continued)

Name	Type	Default	Legal Values
text.blinkcaret	**boolean**	True	True, False
text.checkpointFrequency	**integer**	0	0, <*maxint*>
text.confirmOverwrite	**boolean**	True	True, False
text.displayControlChars	**boolean**	False	True, False
text.enableScrollbar	**boolean**	True	True, False
text.extrasMenuFilename	**string**	*.text_extra_menu* (in */usr/lib*)	*filename*
text.insertMakesCaretVisible	**string**	If_auto_scroll	If_auto_scroll, Always
text.lineBreak	**string**	Wrap_word	Clip, Wrap_char, Wrap_word
text.margin.bottom	**integer**	0	–1, 50
text.margin.left	**integer**	8	0, 2000
text.margin.right	**integer**	0	0, 2000
text.margin.top	**integer**	2	–1, 50
text.maxDocumentSize	**integer**	20000	0, 0x80000000
text.retained	**boolean**	False	True, False
text.storeChangesFile	**boolean**	True	True, False
text.tabWidth	**integer**	8	0, 50
text.undoLimit	**integer**	50	0, 500
window.color.foreground	**string**	"0 0 0"	3 RGB values, 0-255
window.color.background	**string**	"255 255 255"	3 RGB values, 0-255
window.columns	**integer**	80	—
window.height	**integer**	34 rows	Greater than 0
window.header	**string**	NULL	*header string*
window.iconic	**boolean**	False	True, False
window.inheritcolor	**integer**	1	
window.mono.disableRetained	**boolean**	False	True, False
window.rows	**integer**	34	—
window.scale	**string**	Medium	Small, Medium, Large, Extra_Large
window.width	**integer**	80 columns	Greater than 0
window.x	**integer**	0	—
window.y	**integer**	0	—

Note: In XView 3.2 and later, the resource `Font.Name` will still be read by any XView application, however, if `OpenWindows.MonospaceFont`, `OpenWindows.RegularFont` or `OpenWindows.BoldFont` exist in the X Resource Database, they will take precedence.

XView 3.2 and later uses the new resource `OpenWindows.Scale`. In earlier releases of XView, "scale" was determined by reading the resource "Window.Scale." The new resource is synonymous with the old resource name and XView will continue to look at the old resource "Window.Scale" if the new resource, `OpenWindows.Scale`, does not exist. XView provides many functions to get resource values from the database. Basically, values come in several types: `int`, `character` (strings), `boolean` (which can be specified using a number of string values; see below), and `enumerated` values.

Example 17-1 lists a simple application that introduces the use of one of the functions provided by the `defaults` package. The program *default_text.c* creates a frame with a text subwindow whose font is specified by the user's resource `textsw.font`.

Example 17-1. The default_text.c program

```
/*
 * default_text.c -- use the defaults package to get a font name from
 * the resource database to set the textsw's font.
 */
#include <xview/xview.h>
#include <xview/font.h>
#include <xview/defaults.h>
#include <xview/textsw.h>

main(argc, argv)
char *argv[];
{
    Frame       frame;
    Xv_Font     font;
    char        *name;

    xv_init(XV_INIT_ARGC_PTR_ARGV, &argc, argv, NULL);

    frame = (Frame)xv_create(NULL, FRAME, NULL);

    name = defaults_get_string("textsw.font","Textsw.Font", "fixed"),
    font = xv_find(frame, FONT,
        FONT_NAME,      name,
        NULL);

    xv_create(frame, TEXTSW,
        XV_FONT,        font,
        WIN_COLUMNS,    80,
        WIN_ROWS,       10,
        NULL);

    window_fit(frame);
    xv_main_loop(frame);
}
```

default_text.c shows how an XView program queries the resource database for a string value associated with a name-class pair. `defaults_get_string()` gets the string value associated with `Textsw.Font`. If the resource database has this attribute, then the value is returned. If not, the default (`fixed`) is returned.

17.2 XView Resource Database Functions

The functions that XView provides are defined in <*xview/defaults.h*>. The following functions are provided by XView for setting and getting resource values to and from the database. Note that when setting resources *to* the database, the database on the *server* is updated—not the user's *defaults* database. Resources not updated to the user's defaults database are not retained for the next time the X server is started.

```
void
defaults_init_db()
```

This function is called automatically by `xv_init()`, so it need not be called by your application. `defaults_init_db()` calls `XrmInitialize()`.

```
void
defaults_load_db(filename)
    char *filename;
```

`defaults_load_db()` loads the database residing in the specified filename or the server database if filename is `NULL`. The database found in `filename` is loaded via `XrmGetFileDatabase()` and is merged into the existing resource database via `XrmMergeDatabases()`.

```
void
defaults_store_db(filename)
    char *filename;
```

This function writes the database to the specified file via `XrmPutFileDatabase()`. This must be done in order to ensure that the database is accessible the next time the server is started.

```
Bool
defaults_exists(name, class)
    char *name;
    char *class;
```

This function returns `TRUE` if the resource exists in the database via `XrmGetResource()`.

For more information, refer to Volume One, Chapter 11, *Managing User Preferences.*

17.2.1 Boolean Resources

```
Bool
defaults_get_boolean(name, class, default_value)
    char *name, *class;
    int   default_value;

void
defaults_set_boolean(resource, value)
    char *resource;
    Bool  value;
```

`defaults_get_boolean()` looks up the name-class pair in the resource database and returns `TRUE` if the value is one of the following:

- `True`
- `Yes`
- `On`
- `Enabled`
- `Set`
- `Activated`
- `1`

It returns `FALSE` if the value is one of the following:

- `False`
- `No`
- `Off`
- `Disabled`
- `Reset`
- `Cleared`
- `Deactivated`
- `0`

If the value is none of the above, a warning message will be displayed and the default value will be returned. If the resource is not found, no error message is printed but the default value is still returned.

`defaults_set_boolean()` sets the resource to the value specified.

17.2.2 Integer Resources

```
int
defaults_get_integer(name, class, default_value)
    char *name;
    char *class;
    int   default_value;

int
defaults_get_integer_check(name, class, default_value,
                           minimum, maximum)
    char *name;
    char *class;
    int   default_value;
    int   minimum;
    int   maximum;
```

```
void
defaults_set_integer(resource, value)
    char *resource;
    int   value;
```

`defaults_get_integer()` looks up the name-class pair in the resource database and returns the resulting integer value. If the database does not contain the resource, the default value is returned.

`defaults_get_integer_check()` looks up the name-class pair in the resource database and returns the resulting integer value. If the value in the database is not between the values `minimum` and `maximum` (inclusive), an error message is printed and the default value is returned. If the resource is not found, no error message is printed but the default value is returned.

`defaults_set_integer()` sets the resource to the value specified.

17.2.3 Character Resources

```
char
defaults_get_character(name, class, default_char)
    char *name;
    char *class;
    char  default_char;

void
defaults_set_character(resource, character)
    char *resource;
    char  character;
```

`defaults_get_character()` looks up the name-class pair in the resource database and returns the resulting character value. If the resource is not found, then the default character value is returned.

`defaults_set_character()` sets the resource to the character value.

17.2.4 String Resources

```
char *
defaults_get_string(name, class, default_str)
    char *name;
    char *class;
    char *default_str;

void
defaults_set_string(resource, string)
    char *resource;
    char *string;
```

`defaults_get_string()` returns the string value associated with the specified name-class pair in the resource database. If the resource is not found, the default string value is returned. The procedure `defauts_get_string()` returns a pointer into a static buffer maintained by the defaults package. The application should not attempt to free this pointer.

This buffer will be overwritten by the next call to the defaults package, so the application should maintain a copy if necessary.

`defaults_set_string()` sets the resource to the specified string.

17.2.5 Enumerated Resources

Enumerated resources are those whose values are string values, but the legal values for the resource are restricted to a predefined list. For example, say you want to allow the user to specify the font scale for the font `lucidasans`. The legal `scale` values are: small, medium, large, and extra large. You could use `defaults_get_string()` and determine, using `strcmp()`, whether the value returned is one of the legal scale values. Or you could use `defaults_get_enum()` and pass in a table describing the legal values. The *table* is an array of elements of the type:

```
typedef struct _default_pairs {
    char   *name;  /* Name of pair */
    int     value; /* Value of pair */
} Defaults_pairs;
```

The *name* is a string, and the *value* is the returned value associated with the name. This value may be your own value and need not be sequential, but it must be an `int`.

```
int
defaults_get_enum(name, class, pairs)
    char           *name;
    char           *class;
    Defaults_pairs *pairs;
```

`defaults_get_enum()` looks up the value associated with name and class and scans the `pairs` table and returns the associated value. If no match is found, an error is generated and the value associated with the last entry is returned. `defaults_get_enum()` calls `defaults_get_string()` and determines the value returned by calling `defaults_lookup()` (below), passing the returned string as the `name` parameter.

```
int
defaults_lookup(name, pairs)
    char           *name;
    Defaults_pairs *pairs;
```

`defaults_lookup()` linearly scans the `pairs` array looking for `name`. The value associated with `name` is returned. The `pairs` array *must* contain a last element with a `NULL` `name` and a legal `value` associated with it. This value is returned if `name` does not match the `name` field of any of the elements in the `pairs` parameter.

Example 17-2 shows a program that implements the idea discussed earlier for allowing the user to specify a scale for a font.

Example 17-2. The default_size.c program

```
/*
 * default_scale.c -- demonstrate the use of defaults_get_enum().
 * Specify a table of font scales and query the resource database
```

Example 17-2. The default_size.c program (continued)

```
 * for legal values.  For example, you may have the following in
 * your .Xdefaults (which must be loaded into the resource database):
 *      font.scale: large
 */
#include <xview/xview.h>
#include <xview/font.h>
#include <xview/defaults.h>
#include <xview/textsw.h>

Defaults_pairs size_pairs[ ] = {
    "small",            WIN_SCALE_SMALL,
    "medium",           WIN_SCALE_MEDIUM,
    "large",            WIN_SCALE_LARGE,
    "extralarge",       WIN_SCALE_EXTRALARGE,
    /* the NULL entry is the default if Resource not found */
    NULL,               WIN_SCALE_MEDIUM,
};

main(argc, argv)
char *argv[ ];
{
    Frame       frame;
    Xv_Font     font;
    int         scale;

    xv_init(XV_INIT_ARGC_PTR_ARGV, &argc, argv, NULL);

    frame = (Frame)xv_create(NULL, FRAME, NULL);

    scale = defaults_get_enum("font.scale", "Font.Scale", size_pairs);
    /* get the default font for the frame, scaled to resource */
    font = xv_find(frame, FONT,
        FONT_RESCALE_OF,        xv_find(frame, FONT, NULL), scale,
        NULL);

    xv_create(frame, TEXTSW,
        XV_FONT,                font,
        WIN_COLUMNS,            80,
        WIN_ROWS,               10,
        NULL);

    window_fit(frame);
    xv_main_loop(frame);
}
```

In *default_size.c*, the `pairs` table describes an association between string constants and scaling factors. The resource database may specify the resource `font.scale` that describes a scaling factor. If the value in the resource is not one of the legal values included in the table, an error message is printed to inform the user that s/he has specified the resource incorrectly. In this case, or in the case where the resource is not specified, the default resource (associated with the `NULL` entry) is returned, namely, `WIN_SCALE_MEDIUM`.

NOTE

The `value` field in the `Defaults_pair` data structure is an `int`, so do not attempt to assign pointers, functions or attributes to this field. You can assign enumerated types since they are interpreted as `int`.

17.3 Creating Resource Instances

The attribute XV_INSTANCE_NAME is used to associate an instance name with an XView object. The instance name is used to construct the resource name used by the Resource Manager to perform lookups. The resource name is constructed by concatenating the instance names of all objects in the current object's lineage, starting with the name of the application or whatever was passed in with the *–name* command-line option, ending with the XView attribute name. The XView attribute name remains in lowercase. XV_INSTANCE_NAME is normally used with XV_USE_DB. Assume the name of an application is `app`:

```
Frame     frame;
Panel     panel;

frame = (Frame)xv_create(NULL, FRAME,
                        XV_INSTANCE_NAME, "base_frame",
                        NULL);

panel = (Panel)xv_create(frame, PANEL,
                        XV_INSTANCE_NAME, "panel",
                        XV_USE_DB,
                            XV_WIDTH, 100,
                            XV_HEIGHT, 200,
                            NULL,
                        NULL);
```

For this code fragment, assuming the name of the application is "app," the resource names constructed for lookup of the width and height of the panel are:

```
app.base_frame.panel.xv_width
app.base_frame.panel.xv_height
```

Entries in the Resource Manager could look like:

```
app.base_frame.panel.xv_width:400
app.base_frame.panel.xv_height:500
```

If these entries were not present in the Resource Manager, the width and height of the panel would take the default values of 100 and 200 respectively.

The attribute XV_USE_DB specifies a set of attributes that are to be searched in the X11 Resource Manager database. XV_USE_DB takes a NULL-terminated list of attribute value pairs as its values. During program execution, each attribute in this NULL-terminated list of attributes is looked up in the X11 Resource manager database. If the attribute is not found in the database, then the value specified in the attribute-value pair is used as the default value.

The list of customizable attributes can be found in the *XView Reference Manual*.