



*Full credit is given to the above companies including the OS that this PDF file was generated!*

### ***Rocky Enterprise Linux 9.2 Manual Pages on command 'xprop.1'***

**\$ man xprop.1**

XPROP(1)                    General Commands Manual                    XPROP(1)

NAME

xprop - property displayer for X

SYNOPSIS

xprop [-help] [-grammar] [-id id] [-root] [-name name] [-frame] [-font font] [-display display] [-len n] [-notype] [-fs file] [-remove prop?erty-name] [-set property-name value] [-spy] [-version] [-f atom format [dformat]]\* [format [dformat] atom]\*

SUMMARY

The xprop utility is for displaying window and font properties in an X server. One window or font is selected using the command line arguments or possibly in the case of a window, by clicking on the desired window. A list of properties is then given, possibly with formatting information.

OPTIONS

-help Print out a summary of command line options.

-grammar

Print out a detailed grammar for all command line options.

`-id id` This argument allows the user to select window id on the command line rather than using the pointer to select the target window. This is very useful in debugging X applications where the target window is not mapped to the screen or where the use of the pointer might be impossible or interfere with the application.

`-name name`

This argument allows the user to specify that the window named name is the target window on the command line rather than using the pointer to select the target window.

`-font font`

This argument allows the user to specify that the properties of font font should be displayed.

`-root` This argument specifies that X's root window is the target window. This is useful in situations where the root window is completely obscured.

`-display display`

This argument allows you to specify the server to connect to; see X(7).

`-len n` Specifies that at most n bytes of any property should be read or displayed.

`-notype` Specifies that the type of each property should not be displayed.

`-fs file`

Specifies that file file should be used as a source of more formats for properties.

`-frame` Specifies that when selecting a window by hand (i.e. if none of `-name`, `-root`, or `-id` are given), look at the window manager frame (if any) instead of looking for the client window.

`-remove property-name`

Specifies the name of a property to be removed from the indicated window.

`-set property-name value`

Specifies the name of a property and a property value, to be set on the indicated window.

-spy Examine window properties forever, looking for property change events.

-version Print program version information and exit.

-f name format [dformat]

Specifies that the format for name should be format and that the dformat for name should be dformat. If dformat is missing, " = \$0+\n" is assumed.

## DESCRIPTION

For each of these properties, its value on the selected window or font is printed using the supplied formatting information if any. If no formatting information is supplied, internal defaults are used. If a property is not defined on the selected window or font, "not defined" is printed as the value for that property. If no property list is given, all the properties possessed by the selected window or font are printed.

A window may be selected in one of four ways. First, if the desired window is the root window, the -root argument may be used. If the desired window is not the root window, it may be selected in two ways on the command line, either by id number such as might be obtained from xwininfo, or by name if the window possesses a name. The -id argument selects a window by id number in either decimal or hex (must start with 0x) while the -name argument selects a window by name.

The last way to select a window does not involve the command line at all. If none of -font, -id, -name, and -root are specified, a crosshairs cursor is displayed and the user is allowed to choose any visible window by pressing any pointer button in the desired window. If it is desired to display properties of a font as opposed to a window, the -font argument must be used.

Other than the above four arguments and the -help argument for obtaining help, and the -grammar argument for listing the full grammar for the command line, all the other command line arguments are used in

specifying both the format of the properties to be displayed and how to display them. The `-len n` argument specifies that at most `n` bytes of any given property will be read and displayed. This is useful for example when displaying the cut buffer on the root window which could run to several pages if displayed in full.

Normally each property name is displayed by printing first the property name then its type (if it has one) in parentheses followed by its value. The `-notype` argument specifies that property types should not be displayed. The `-fs` argument is used to specify a file containing a list of formats for properties while the `-f` argument is used to specify the format for one property.

The formatting information for a property actually consists of two parts, a format and a dformat. The format specifies the actual formatting of the property (i.e., is it made up of words, bytes, or longs?, etc.) while the dformat specifies how the property should be displayed. The following paragraphs describe how to construct formats and dformats. However, for the vast majority of users and uses, this should not be necessary as the built in defaults contain the formats and dformats necessary to display all the standard properties. It should only be necessary to specify formats and dformats if a new property is being dealt with or the user dislikes the standard display format. New users especially are encouraged to skip this part.

A format consists of one of 0, 8, 16, or 32 followed by a sequence of one or more format characters. The 0, 8, 16, or 32 specifies how many bits per field there are in the property. Zero is a special case meaning use the field size information associated with the property itself. (This is only needed for special cases like type INTEGER which is actually three different types depending on the size of the fields of the property.)

A value of 8 means that the property is a sequence of bytes while a value of 16 would mean that the property is a sequence of words. The difference between these two lies in the fact that the sequence of words will be byte swapped while the sequence of bytes will not be when

read by a machine of the opposite byte order of the machine that originally wrote the property. For more information on how properties are formatted and stored, consult the Xlib manual.

Once the size of the fields has been specified, it is necessary to specify the type of each field (i.e., is it an integer, a string, an atom, or what?) This is done using one format character per field. If there are more fields in the property than format characters supplied, the last character will be repeated as many times as necessary for the extra fields. The format characters and their meaning are as follows:

- a The field holds an atom number. A field of this type should be of size 32.
- b The field is a boolean. A 0 means false while anything else means true.
- c The field is an unsigned number, a cardinal.
- i The field is a signed integer.
- m The field is a set of bit flags, 1 meaning on.
- o The field is an array of icons, packed as a sequence of 32 bit numbers consisting of the width, height and ARGB pixel values, as defined for the `_NET_WM_ICON` property in the Extended Window Manager Hints specification. A field of this type must be of size 32.
- s This field and the next ones until either a 0 or the end of the property represent a sequence of bytes. This format character is only usable with a field size of 8 and is most often used to represent a string.
- t This field and the next ones until either a 0 or the end of the property represent an internationalized text string. This format character is only usable with a field size of 8. The string is assumed to be in an ICCCM compliant encoding and is converted to the current locale encoding before being output.
- u This field and the next ones until either a 0 or the end of the property represent a UTF-8 encoded unicode string. This format character is only usable with a field size of 8. If the string

is found to be an invalid character, the type of encoding violation is printed instead, followed by the string formatted using 's'. When in an environment not capable of displaying UTF-8 encoded string, behaviour is identical to 's'.

x The field is a hex number (like 'c' but displayed in hex - most useful for displaying window ids and the like)

An example format is `%32ica` which is the format for a property of three fields of 32 bits each, the first holding a signed integer, the second an unsigned integer, and the third an atom.

The format of a `dformat` unlike that of a `format` is not so rigid. The only limitations on a `dformat` is that one may not start with a letter or a dash. This is so that it can be distinguished from a property name or an argument. A `dformat` is a text string containing special characters instructing that various fields be printed at various points in a manner similar to the formatting string used by `printf`. For example, the `dformat` " is ( \$0, \$1 )\n" would render the POINT 3, -4 which has a format of `%32ii` as " is ( 3, -4 )\n".

Any character other than a \$, ?, \, or a ( in a `dformat` prints as itself. To print out one of \$, ?, \, or ( precede it by a \. For example, to print out a \$, use `\$`. Several special backslash sequences are provided as shortcuts. `\n` will cause a newline to be displayed while `\t` will cause a tab to be displayed. `\o` where o is an octal number will display character number o.

A \$ followed by a number n causes field number n to be displayed. The format of the displayed field depends on the formatting character used to describe it in the corresponding format. I.e., if a cardinal is described by 'c' it will print in decimal while if it is described by a 'x' it is displayed in hex.

If the field is not present in the property (this is possible with some properties), `<field not available>` is displayed instead. `$n+` will display field number n then a comma then field number n+1 then another comma then ... until the last field defined. If field n is not defined, nothing is displayed. This is useful for a property that is a

list of values.

A `?` is used to start a conditional expression, a kind of if-then statement. `?exp(text)` will display text if and only if `exp` evaluates to non-zero. This is useful for two things. First, it allows fields to be displayed if and only if a flag is set. And second, it allows a value such as a state number to be displayed as a name rather than as just a number. The syntax of `exp` is as follows:

```
exp ::= term | term=exp | !exp
```

```
term ::= n | $n | mn
```

The `!` operator is a logical "not", changing 0 to 1 and any non-zero value to 0. `=` is an equality operator. Note that internally all expressions are evaluated as 32 bit numbers so -1 is not equal to 65535. `=` returns 1 if the two values are equal and 0 if not. `n` represents the constant value `n` while `$n` represents the value of field number `n`. `mn` is 1 if flag number `n` in the first field having format character 'm' in the corresponding format is 1, 0 otherwise.

Examples: `?m3(count: $3\n)` displays field 3 with a label of count if and only if flag number 3 (count starts at 0!) is on.

`?$2=0(True)?!$2=0(False)` displays the inverted value of field 2 as a boolean.

In order to display a property, `xprop` needs both a format and a `dformat`. Before `xprop` uses its default values of a format of `32x` and a `dformat` of `" = { $0+ }\n"`, it searches several places in an attempt to find more specific formats. First, a search is made using the name of the property. If this fails, a search is made using the type of the property. This allows type `STRING` to be defined with one set of `formats` while allowing property `WM_NAME` which is of type `STRING` to be defined with a different format. In this way, the display formats for a given type can be overridden for specific properties.

The locations searched are in order: the format if any specified with the property name (as in `8x WM_NAME`), the formats defined by `-f` options in last to first order, the contents of the file specified by the `-fs` option if any, the contents of the file specified by the `environmental`

variable XPROPFORMATS if any, and finally xprop's built in file of formats.

The format of the files referred to by the -fs argument and the XPROPFORMATS variable is one or more lines of the following form:

name format [dformat]

Where name is either the name of a property or the name of a type, format is the format to be used with name and dformat is the dformat to be used with name. If dformat is not present, " = \$0+\n" is assumed.

## EXAMPLES

To display the name of the root window: xprop -root WM\_NAME

To display the window manager hints for the clock: xprop -name xclock WM\_HINTS

To display the start of the cut buffer: xprop -root -len 100 CUT\_BUFFER0

To display the point size of the fixed font: xprop -font fixed POINT\_SIZE

To display all the properties of window # 0x200007: xprop -id 0x200007

To set a simple string property: xprop -root -format MY\_ATOM\_NAME 8s -set MY\_ATOM\_NAME "my\_value"

## ENVIRONMENT

DISPLAY To get default display.

### XPROPFORMATS

Specifies the name of a file from which additional formats are to be obtained.

## SEE ALSO

X(7), xdpinfo(1), xwininfo(1), xdriinfo(1), glxinfo(1), xvinfo(1)

## AUTHOR

Mark Lillibridge, MIT Project Athena

X Version 11

xprop 1.2.3

XPROP(1)