ACORN R140

RISC *i*X X.desktop GUIDE

3034/21382

Acorn

The choice of experience.

Copyright © Acorn Computers Limited 1988

X.desktop has been developed by IXI Limited

Neither the whole nor any part of the information contained in, nor the product described in this manual may be adapted or reproduced in any material form except with the prior written approval of Acorn Computers Limited.

The product described in this manual and products for use with it are subject to continuous development and improvement. All information of a technical nature and particulars of the product and its use (including the information and particulars in this manual) are given by Acorn Computers Limited in good faith. However, Acorn Computers Limited cannot accept any liability for any loss or damage arising from the use of any information or particulars in this manual.

All correspondence should be addressed to:
Customer Support and Service
Acorn Computers Limited
Fulbourn Road
Cherry Hinton
Cambridge
CB1 4JN

X.desktop, IXI, and the IXI logo are trademarks or registered trademarks of IXI Limited.
ACORN is a trademark of Acorn Computers Limited.
UNIX is a trademark of AT&T.

Published December 1988: Issue A
ISBN 1 85250 059 X
Published by Acorn Computers Limited

Part Number 0483,749

COMPUTING DEPT

Contents

Part 1: X.desktop User Guide

Part 2: X.desktop Configuration

Section 1: Introduction

Section 2: Tutorials

Section 3: Reference

Part 1 – X.desktop User Guide

About the X.desktop User Guide

This guide assumes that you have logged in to your RISC iX workstation as user 'desktop', as detailed in the *R140 Operations Guide*. This guide tells you how to use X.desktop. It's a short guide, because X.desktop is so easy to use.

This guide is for all new X.desktop users. X.desktop runs on RISC iX workstations, but this guide doesn't assume you're familiar with UNIX or C. The person who supplied your system may have added some extra features, but this guide covers all features in the standard distribution of X.desktop.

However, the guide does assume that you're familiar with using a mouse and keyboard, and with the idea of different types of computer files. See the *R140 Operations Guide* for an introduction to the R140 keyboard and mouse.

The guide is organised into short sections covering different aspects of using X.desktop. This is to help you work quickly through the guide and learn how to use X.desktop as you go along.

Conventions used in this guide

This guide uses several different conventions, and this section explains them.

A numbered list gives you a sequence of instructions to follow, like this:

1 start this

2 do that

3 finish.

Menus and menu options look like this:

Put back

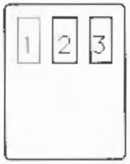
Comments for the more experienced UNIX user are shown like this:

❖ You can also use this by doing that. ◆

This guide has instructions for using the mouse, which are listed below:

Instruction	What you do
Select	Click once.
Activate	Click twice in quick succession.
Drag	Press and hold the button down, move the mouse, and let go.
Get	Click once on the desktop background or on the directory background.

Your mouse has three buttons. Use the leftmost button (button 1) unless we say otherwise. X. desktop normally only uses the left and centre buttons, but your system may be configured to use the right button (button 3).



Getting further information

If you have any queries, contact your System Administrator or your support organisation.

Introducing X.desktop

X.desktop is an easy way to use RISC iX workstations. It lets you exploit the full potential of your workstation by giving you an immediate, visual system which helps you manage your work efficiently, no matter what your level of UNIX experience is.

X.desktop and RISC iX

With X.desktop, you see the screen as a desktop, which displays programs and files for you to work with. Some of these files are always on the desktop. You can put more items on it as you need them, use them, and put them away.

X.desktop shows each file as a picture, called an icon, on the desktop. These icons replace the lists of directories' contents that ordinary UNIX gives you.

The X.desktop approach

Because you can see your files on the desktop, you have a valuable way of managing your work. X.desktop gives you an organisational environment that makes it easy for you to manage your work effectively.

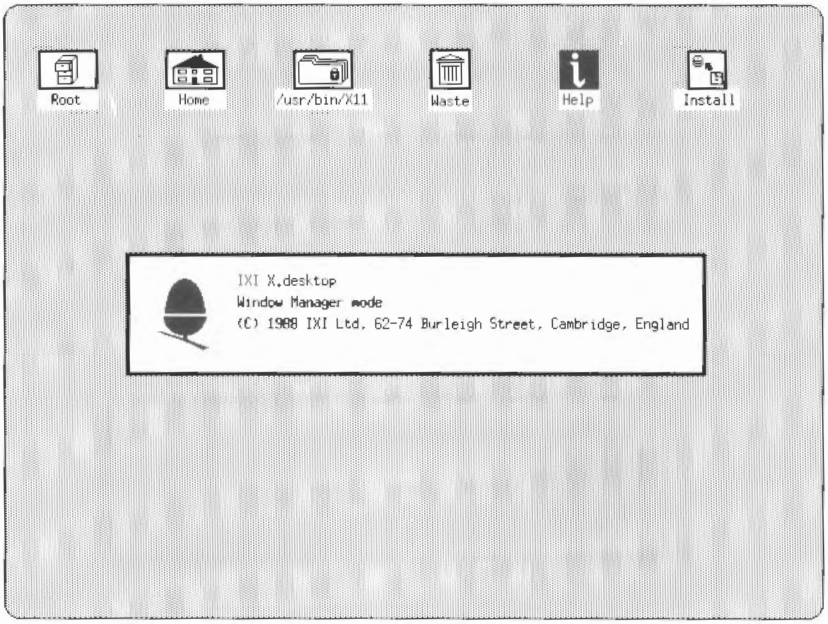
And you don't have to type any commands. With X.desktop, you can be a UNIX user without knowing a single UNIX command. You have easy access to standard UNIX functions using a mouse with the icons on the screen.

You can see what you're doing because you can put all the files you need on the desktop. It's very quick to use – and you can't make typing mistakes when you use the mouse.

❖ If you're an experienced UNIX user, you have all these benefits plus access to a UNIX shell whenever you need one. ◆

Using your desktop

This section begins at the point when X.desktop is up and running on your workstation. The desktop, on which you set out and use RISC iX programs, looks like this, although the icons and the wording may be slightly different:



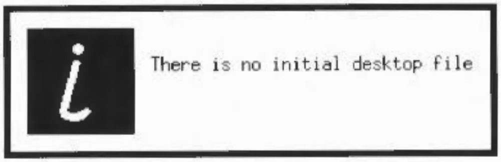
IXI X.desktop
Window Manager mode
(C) 1988 IXI Ltd, 62-74 Burleigh Street, Cambridge, England

The box in the middle with the Acorn logo is an information window.

Note: If your screen goes blank, your R140 screen may have been configured to do this if you do not use the keyboard for a certain length of time. Pressing any key will restore the screen in this case.

Information windows

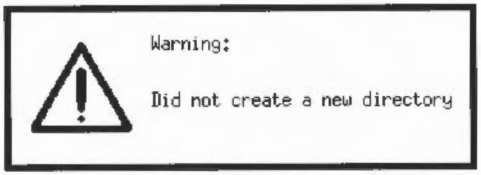
As you use X.desktop, you may see more information windows with symbols replacing the Acorn logo. One type has the symbol 'i' for information.



There is no initial desktop file

When you've read such a window, you should make it disappear so that you can carry on with X.desktop. To do this, click on the logo or symbol at the left of the window.

There are two other types of information window – a warning window and a stop window:



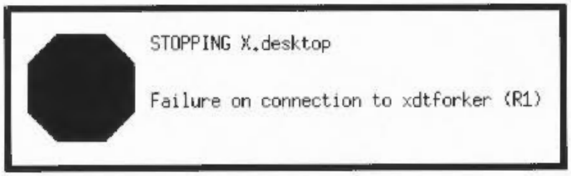
Warning:

Did not create a new directory

A triangle means it's a warning window, telling you something you need to know about. The message explains what's happening and what you should do.

Remember to click on the symbol or you will not be able to continue.

In the unlikely event of something affecting X.desktop so that it can't carry on, you'll see the stop window.



STOPPING X.desktop

Failure on connection to xdtforker (R1)

This means that X.desktop is letting you know what it's doing.

Remember to click on the symbol or you will not be able to continue.

Using the icons

When you start X.desktop, some icons are already out on the desktop.

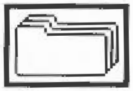
Your system is configured so that certain icons are always there. These are called fixed icons – they're fixed on the desktop because without them you couldn't use X.desktop. For example, without the main directory, you couldn't reach any of the other files. There's more about fixed icons on page 20.

Each time you go into X.desktop, you see the icons you left on the desktop the last time you used it. This means that the files you use most often are ready for you to use as soon as you start the system.

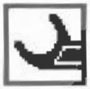
What the icons look like

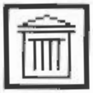
Each icon represents a different type of file.

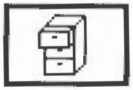
The main icons you use are shown below:

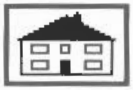

directories, which contain other files and directories

data files


programs

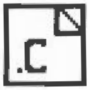

the Waste directory


root

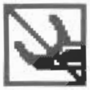

home

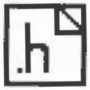
The text beneath each icon is its title. This is usually the name of the file or directory. It can be different if your system rules are set up that way – for example, all your personal files could use your initials followed by the filename as the icon title.

There are many other icons which you'll see from time to time although you may not use them all, such as the ones below.

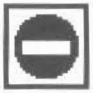


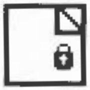
C program files

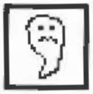

programs you can't run

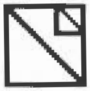


C header files

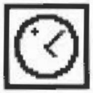

files the computer won't tell you about


data files you can't change

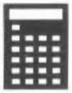

ghost files – there's more about these on page 26


data files you can't look at

There may be others for other programs, such as a clock and a calculator.

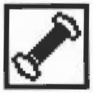


clock

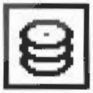


calculator

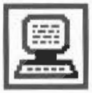
❖ There are icons for the special RISC iX files:



pipes



disc



terminal control files

Moving icons

You move icons around on the desktop, putting them where it suits you. To move an icon:

1 put the cursor on the icon picture (**take care not to click on the title**)

2 drag the icon where you want (see page 4).

When you drag an icon, the cursor changes shape and becomes a mini-icon, to show that you've successfully picked up the icon. This moves as you drag, to show you where you are on the desktop. When you let go of the mouse button, the item you've dragged moves to its new place.

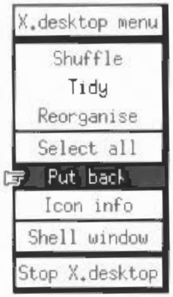
Using the menus

X.desktop has two menus: the **X.desktop menu** and the **Directory menu**, which have different options. This guide covers all the options on both these menus.

To get the X.desktop menu:

Click anywhere on the desktop background.

On a menu, the cursor changes to a hand, pointing at the appropriate option.



To choose an option:

Click on the one you want.

Or instead, you can:

1 press and hold on the desktop to get the **X.desktop menu**

2 move the cursor to the option you want – the highlighting shows you where you are

3 let go.

For example:

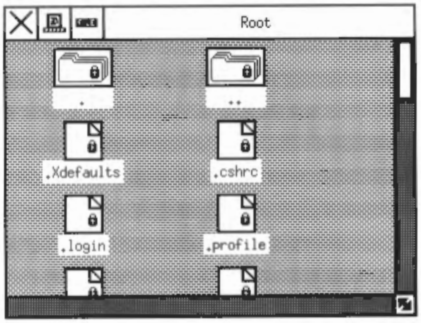
- drag an icon about half an inch from its place (see page 4)
- get the **X.desktop menu**
- choose **Tidy** (release the mouse button when you have highlighted **Tidy**).

To remove the menu:

Click on the title.

To see the Directory menu:

1 open any directory, by activating a directory icon. (see page 4)

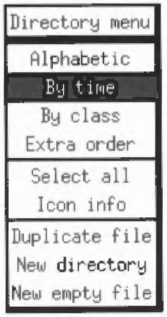


There's more detail about directories and their windows later, on pages 17 to 20.

You may have to wait a short while for the contents of the directory to appear – if, for example, you're using X.desktop on a network or a multi-user system. The cursor becomes an hourglass, which tells you to wait.

When the hourglass disappears, to see the menu:

2 click on the background of the open directory.



Use this like the **X.desktop menu**. The highlighting shows you where you are.

Getting icon information

You can look at information about the file that an icon represents. **Icon info** appears on both the X.desktop menus.

To look at the information:

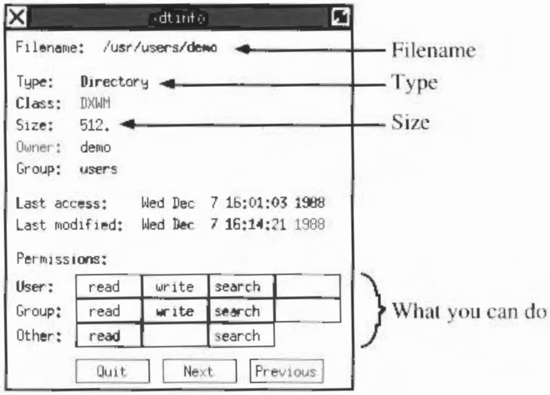
1 select the icon (see page 4) – a black box appears around the title, to show that the icon's been selected

2 get either menu (see page 4)

3 choose **Icon info**.

You see a window that shows you:

- the full filename of the icon
- the file access permissions
- the size of the file.



```
Filename:  /usr/users/demo

Type:     Directory
Class:    DXWM
Size:     512.
Owner:    demo
Group:    users

Last access:     Wed Dec  7 16:01:03 1988
Last modified:   Wed Dec  7 16:14:21 1988

Permissions:
```

User:	read	write	search	
Group:	read	write	search	
Other:	read		search	

To make the window disappear:

Click on **Quit**.

Icon info is a program, so you can also use the stop box – there's more about this on page 22.

You can also use **Select all** from either menu, and then **Icon info**. You see information for all the icons you've selected, one after the other. To move on to the next icon, click on **Next**.

Organising the desktop

X.desktop lets you organise things the way you want to as you're working.

Shuffling directory windows

You can open as many directories as you need to at the same time, and it's easy to keep track of what you're doing. X.desktop automatically stacks them behind each other, and you bring them to the front as you need to use them.

To bring a directory window to the front:

Click on any part of it you can see.

If you can't see it because it's completely hidden behind another window:

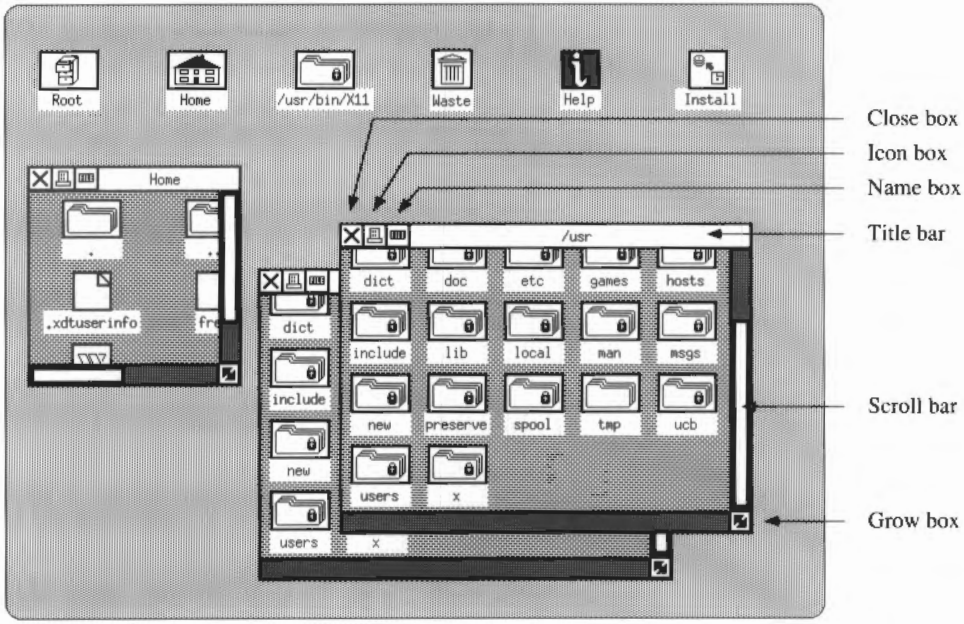
1 get the **X.desktop menu**

2 choose **Shuffle** – whatever is at the back comes to the front.

If there are several directory windows stacked, you may need to choose **Shuffle** more than once to get the one you want.

Moving windows on the desktop

You often need to look at several windows at the same time, for example if you're using files in different directories. You can move the windows around and change their size to fit them on the desktop.



To move a window:

Drag the title bar (see page 4).

To change a window size:

Drag the grow box (see page 4).

When you change a window, X.desktop changes the length of the scroll bars to fit. If you want to look at files that don't fit in the window, and you don't want to make the window larger, you drag the scroll bars to see the files. You can't make a window smaller than the size of one icon, or bigger than the desktop.

Tidying up

To close a directory window:

Click on the close box.

To tidy up the desktop:

1 get the **X.desktop menu** (see page 4)

2 choose **Tidy**.

This moves each icon to the nearest available place on an invisible grid, stacking them on top of each other if necessary.

To rearrange the icons:

1 get the **X.desktop menu** (see page 4)

2 choose **Reorganise**.

X.desktop rearranges the icons in rows, starting across the top of the desktop.

Managing directories

X.desktop lets you see at a glance the types and number of files that RISC iX directories contain.

When you look at the files, they're arranged into directories. Directories can contain directories, which can contain directories, and so on. This means that you can group files together for convenience – for example, according to the sort of information they contain or who uses them.

A file's full name tells you where it is in the hierarchy of directories. For example, if you have a file named `tadpole`, `tadpole` is the file's basename. This file is in a directory named `frog` – its parent directory. The directory `frog` is in a directory named `pond`, which is in the main, or root directory. So the full name, or pathname of the file `tadpole` is: `/pond/frog/tadpole`.

There's more about this in the section *Working in different directories*.

Looking at the contents

You can choose how you look at what's in a directory, according to what's most convenient.

To open the directory:

Activate the directory icon.

You see the files in the directory. The window also contains two icons with `.` and `..` as their titles. The icon `.` is the directory itself and `..` is its parent directory. You can drag these icons on to the desktop, and you can select them individually and then get **Icon info** about the directory you're working in.

❖ `.` and `..` have their usual UNIX functions. ◆

Name or icon

X.desktop usually shows the contents of a directory as icons with their titles, rather than the real filenames, but you can change this.

To look at the files by name:

Click on the directory's file box.

To look at the files as icons with titles again:

Click on the directory's icon box.

When you do either of these, X.desktop also sorts the files according to the order that's currently selected.

Choosing the order

You can show the files in a directory according to the following orders:

- alphabetical
- the time they were changed, with the most recently changed first
- the class of file – for example, directory, program, data.

Your system is set so that it defaults to one of these.

To look at the contents in another way:

1 get the **Directory menu**

2 choose the order you want.

There's also the option **Extra order** on the menu. This can be set up when your system is configured, as an extra way of ordering the directory.

Creating a new directory

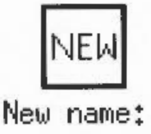
If you want to group files together – for example, if you're starting to work on a new project – you can create a new directory at any level in the structure, to put together your new files.

To create a new directory:

1 open the directory in which you want to put the new one

2 get the **Directory menu**

3 choose **New directory** – a blank icon appears at the bottom of the window, and X.desktop scrolls the window so that you can see it:



New name:

4 type the new name then press <↵> – you can use letters, digits, dashes, dots and the underline character.

You have to name the new directory – if you try to do anything before you've done this, you lose the blank icon and X.desktop gives you a warning window with the message:

```
Did not create a new directory
```

Working in different directories

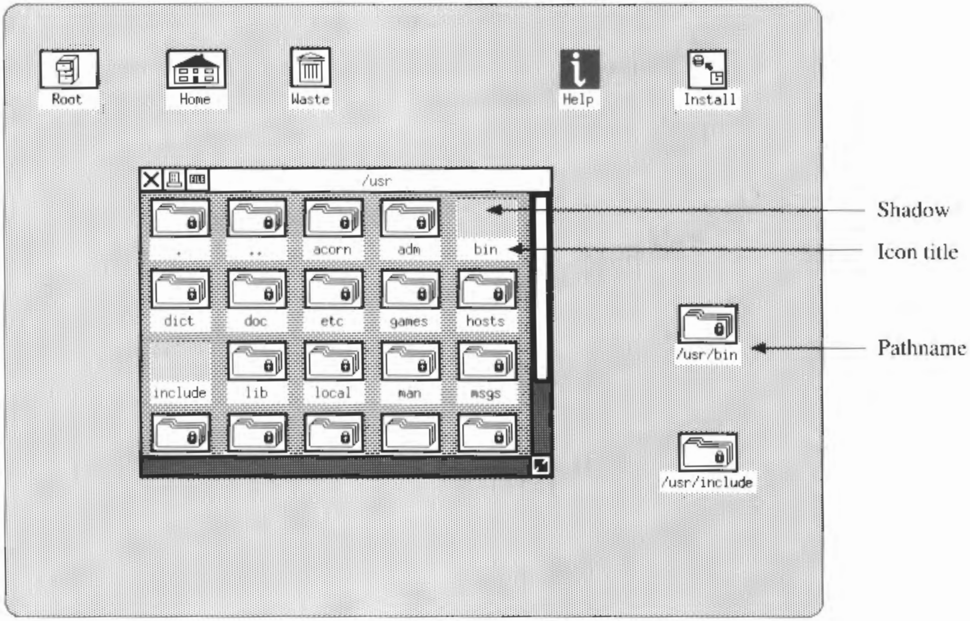
As you work, you use some files more frequently than others. They may be in different directories, but with X.desktop it's easy to switch between them. You can take icons out of their directories and keep them ready for use on the desktop.

To get out an icon:

1 open the directory

2 drag the icon on to the desktop.

The icon leaves a shadow in the directory to show that it's been taken out.



If the icon title is the filename, when the icon's out on the desktop the title changes to the file's pathname, showing where the file fits in the directories' structure.

When you close the directories, the icons stay on the desktop. You don't have to open the windows again to put them back.

To put back an icon:

Drag the icon back into the directory, if it's open.

Or:

1 select the icon

2 get the **X.desktop menu**

3 choose **Put back** and the icon disappears from the desktop.

When you're tidying up, you can put back everything on the desktop at once:

1 get the **X.desktop menu**

2 choose **Select all**

3 choose **Put back**.

Only the fixed icons stay on the desktop – there's no way of putting these back.

You may be able to select more than one icon without selecting them all, if your system's been set up that way: use the second mouse button to select the icons after the first one.

Running programs

This section explains how you can run programs from X.desktop, even if you don't know UNIX.

Starting a program

In X.desktop, you don't need to give long and complicated commands to run programs.

To run a program:

Activate the program icon.

To run your program with a data file:

Drag the data file to the program.

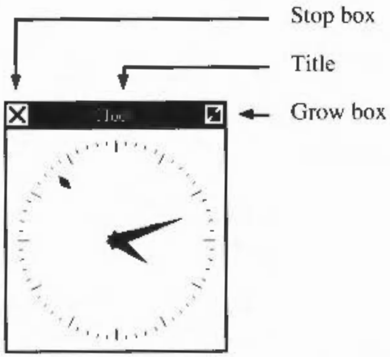
You see the cursor change to a lightning flash to show that X.desktop is running the program.

Starting an application may take some time – the amber H/DISC activity light may flash as the software is loaded from the disc.

Some programs may not accept data files, and other programs may only run with data files, depending on how your system is set up. It may be set up so that when you activate a data file, X.desktop knows which program to run. For example, if you activate a text file, X.desktop runs it with a word processor or editor. Exactly which program X.desktop chooses depends on how the system is set up.

Process windows

In X.desktop, programs run inside a process window, like **Icon info**, for example, or the clock program shown below.



These are different from directory windows in the following ways:

- the grow box is at the top righthand corner
- you can stack and **Shuffle** them with directory windows, but to bring a process window to the front you can also click on its title bar
- when the window is at the front, the title bar is highlighted, and anything you type then goes into that window. When windows are side-by-side, activate a window by clicking on its title bar.

Closing a process window

You can leave a program running, but close the process window, so that your desktop doesn't get cluttered. To do this, click on the stop box.

You then see a process icon. This represents a running program and not a file like the other icons, so you can't select it or drag it to a data file. However, you can move it around the desktop like any other icon.

To see the process window again:

Click once on the process icon.

Stopping a program

Every program has its own commands. To stop a program:

1 move the cursor into the process window

2 use the program's stop command.

However, X.desktop gives you an extra facility that you can use as a last resort if you have to stop a program and you don't know the command.

To use this facility, double-click on the stop box – the window disappears and the program stops running.

❖ Using UNIX commands

If you need to use UNIX commands rather than the desktop, you can have a UNIX shell in a window. You can have more than one shell running at the same time.

A quick way to run the shell is:

1 get the **X.desktop menu**

2 choose **Shell window**, and it appears over the desktop like any other process.

The shell is now ready to use. **Stop the shell like any other program, with its own command or double-clicking on the stop box.** ◆

Managing files

With X.desktop, you can easily keep track of all your files.

Creating a new file

You create a new, empty file in the same way as a new directory (see *Creating a new directory*):

1 open the directory in which you want to put the new file

2 get the **Directory menu**

3 choose **New empty file** – a blank icon appears at the bottom of the window, and X.desktop scrolls so that you can see it

4 type the new name then press <↵> – you can use letters, digits, dashes, dots and the underline character.

You have to name the new file – if you try to do anything before you've done this, you lose the blank icon and see a warning window with the message:

```
Did not create a new file
```

Copying

You can copy files from one directory to another.

To copy a file:

1 open the directory windows you need

2 drag the icon to the new directory.

Don't let go on top of another icon, because that would drop the icon and tell X.desktop to run a program.

The new icon appears at the end of the directory and, if necessary, X.desktop scrolls the window so that you can see it.

If you try to copy a file into a directory that already has a file with the same basename, you see a warning window containing the message:

```
Can't copy
```

followed by the names of the file and directory.

Duplicating

You may want to have a duplicate file, for example so that you have a copy of the previous version to return to should you need it. Duplicating a file is, in effect, copying within the same directory. Because the new file's in the same directory, it can't have the same basename, so X.desktop asks you for another.

To create a duplicate file:

1 open the directory

2 select the file you want to duplicate

3 get the **Directory menu**

4 choose **Duplicate file** – a blank icon appears at the bottom of the window, and X.desktop scrolls so that you can see it

5 name the duplicate file, in the same way as you do a new, empty file.

Renaming

You may want to rename files or directories so that, for example, you can:

- change the way you group them together as your work develops
- change the file extension.

To do this:

1 select the icon title – you see the message `new name:` and the current basename

2 use <Backspace> or <Delete> to erase part or all of the old name

3 type the new name and press <↵>.

Deleting

To delete a file, drag the icon on to the Waste icon – the icon disappears from the desktop and its directory.

If someone has deleted a file from your desktop since you last used X.desktop, there is a ghost icon with the title of the deleted file. A ghost icon shows that X.desktop is looking for something that isn't there. You may also see a ghost icon if a program deletes a file while it's on the desktop, and tells X.desktop that it has done so.

Put back removes the ghost icon from the desktop. Because the icon doesn't represent a file, it doesn't actually go back into a directory – it just disappears.

Moving

You can use the second mouse button to move files from one directory to another.

To move a file:

1 open the directory windows

2 drag the icon to the new directory, using the second button.

Glossary

basename
A file's name within its own directory.

close box
The box at the top left of a directory window, with a cross in it, on which you click to remove the window.

grow box
The box at the bottom right of a directory window or the top right of a process window, which you drag to change the window's size.

hierarchy
The 'tree' structure of directories. They all come from the main, or 'root' directory.

home directory
A directory which you always work in, if your system's set up that way. This means you don't have to go through a series of directories to get to the one you want.

information window
A special type of window, giving you information and warnings about the system, which you should note before you carry on.

parent directory
A file's parent directory is the directory that contains it.

pathname
A file's name, showing by directory names how to get from the main directory to that file.

root directory
The main directory, which contains all the others – it's also called /.

scroll bars
The horizontal or vertical bars in a directory window's border. You drag them along the border to scroll the files and see what's beyond the edge of the window.

shell
An all-purpose command interpreter, which is the usual starting point for running programs if you don't have X.desktop.

stop box
The box at the top left of a program window, with a cross in it, on which you click once to turn the window into an icon, and twice to stop the program.

System Administrator
The person who manages your workstation and/or network.

title bar
The banner at the top of a window, also used for moving it.

window
A rectangular area on the desktop, representing an open directory or a program that's running, or containing a message from X.desktop.

Glossary

basename
A file's name within its own directory.

close box
The box at the top left of a directory window, with a cross in it, on which you click to remove the window.

grow box
The box at the bottom right of a directory window or the top right of a process window, which you drag to change the window's size.

hierarchy
The 'tree' structure of directories. They all come from the main, or 'root' directory.

home directory
A directory which you always work in, if your system's set up that way. This means you don't have to go through a series of directories to get to the one you want.

information window
A special type of window, giving you information and warnings about the system, which you should note before you carry on.

parent directory
A file's parent directory is the directory that contains it.

pathname
A file's name, showing by directory names how to get from the main directory to that file.

root directory
The main directory, which contains all the others – it's also called /.

scroll bars
The horizontal or vertical bars in a directory window's border. You drag them along the border to scroll the files and see what's beyond the edge of the window.

shell
An all-purpose command interpreter, which is the usual starting point for running programs if you don't have X.desktop.

stop box
The box at the top left of a program window, with a cross in it, on which you click once to turn the window into an icon, and twice to stop the program.

System Administrator
The person who manages your workstation and/or network.

title bar
The banner at the top of a window, also used for moving it.

window
A rectangular area on the desktop, representing an open directory or a program that's running, or containing a message from X.desktop.

Part 2 – X.desktop – Configuration

About X.desktop Configuration

This part of the guide, *X.desktop Configuration*, describes the structure of the rule and defaults files that determine the behaviour and appearance of X.desktop, and gives information on how these can be altered to suit particular applications. The guide is divided into three sections.

Section 1, *Introduction*, gives a general introduction to rule and defaults files, and describes some typical applications in which they can be used.

Section 2, *Tutorials*, presents a series of four tutorials in each of which a specific application is described in detail.

Section 3, *Reference*, gives a precise definition of the components of rule and defaults files, together with the syntax of these components.

Conventions used in this guide

The following conventions are used for describing the syntax of the components of configuration files in this guide. For example, in:

Syntax: `abc { `*`file-name`*` [ , `*`file-spec`*` ] … }`

items in computer typeface, such as `abc`, represent text you would type into the computer;

items in italics, such as *`file-name`*, represent a class of object;

items in square brackets, such as `[ , `*`file-spec`*` ]`, are optional;

ellipsis '…' indicates that the previous object can be repeated any number of times.

This guide assumes that you are already familiar with the default operation of X.desktop, which is described in the *X.desktop User Guide*. It also assumes that you are familiar with UNIX.

Note: In this guide, the term UNIX is used to refer both to UNIX in general and to Acorn's implementation, RISC iX.

The publications referred to in this guide may be obtained from your supplier, or direct from IXI Limited, 62-74 Burleigh Street, Cambridge CB1 1OJ, England.

Section 1
Introduction

Configuring X.desktop

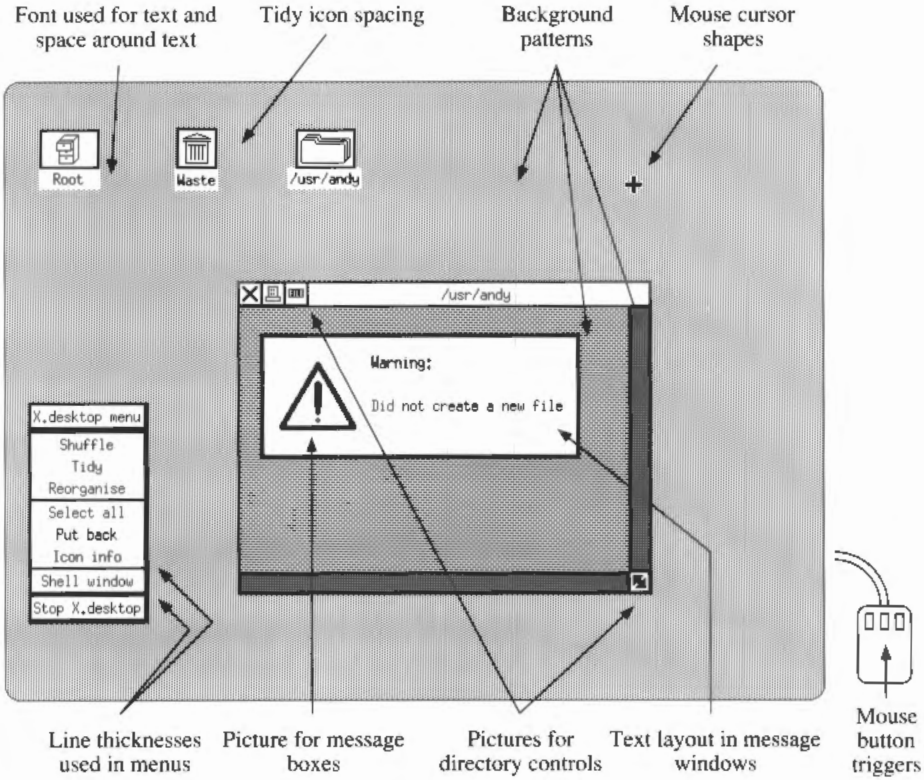
As a user of X.desktop, you are probably familiar with the appearance of the desktop and its components, the file and directory icons, the menus, the directory windows, the information windows, and so on. You are also certainly familiar with the way in which you manipulate these components with the mouse and keyboard to carry out actions on files and programs in a far more convenient and friendly way than UNIX itself provides.

The most powerful feature of X.desktop is that its behaviour and appearance is not fixed, but is determined by rule files which can be edited to suit individual requirements.

This guide explains how you can change the underlying appearance and behaviour of X.desktop to suit your own preferences, or specific applications not catered for by the default configuration.

Changing the appearance of X.desktop

The defaults files determine the default appearance of the main components of X.desktop, as shown in Fig. 1.



Warning:

Did not create a new file

Fig.1: Characteristics of X.desktop specified by defaults files.

In addition, the defaults files specify the mapping between the mouse clicks and presses, and the triggers used by the system. These can be altered to cater for a mouse with a different number or layout of buttons.

The system defaults file specifies the start-up environment file, which contains the desktop layout. This file is updated each time you leave X.desktop.

Normally each computer running X.desktop will have a defaults file suited to the particular machine on which it is being run. For example, machines with large screens will use larger pictures for window controls to improve legibility. However, users can provide their own defaults files to give any desired appearance, or can switch between defaults files to provide different working environments for different applications.

Changing the behaviour of X.desktop

A separate set of files, called rule files, determine the aspects of X.desktop shown in Fig. 2.

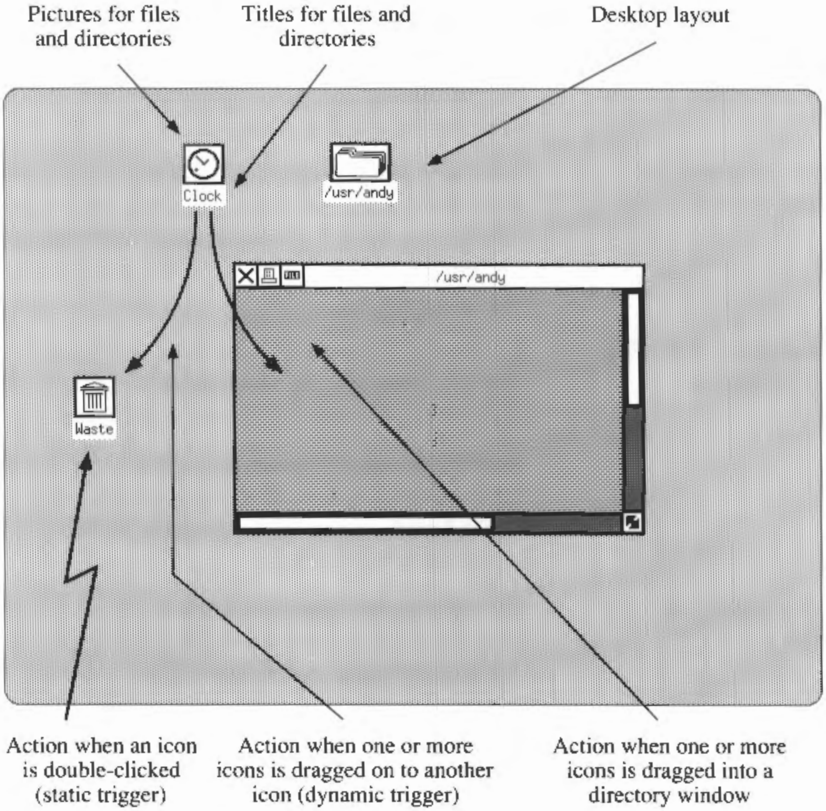

Pictures for files and directories
Titles for files and directories
Desktop layout
Clock
/usr/andy
/usr/andy
Waste
Action when an icon is double-clicked (static trigger)
Action when one or more icons is dragged on to another icon (dynamic trigger)
Action when one or more icons is dragged into a directory window

Fig. 2: Characteristics of X.desktop specified by rule files.

The default behaviour of X.desktop is determined by a system rule file, but this can be overridden by additional rule files provided by each user. Furthermore, rule files can be created which are local to a specific directory.

Rule files consist of a number of separate components which determine the following aspects of the behaviour of X.desktop:

Icon appearance and title – the picture displayed for a file or group of files, and the title displayed below it.

Icon activation – what happens when the icon is activated, or double-clicked, by the mouse, and what happens when another icon is dropped on to it.

Drop behaviour – what happens when one or more icons is dragged into a directory window.

Desktop layout – which files are on the desktop, and what their positions are.

Locked files – which files are permanently locked on the desktop.

Typical applications

The following examples illustrate how X.desktop can be configured in some particular practical applications.

Simulating familiar environments

Users who are already familiar with other desktop-based systems, or who have to share their work between X.desktop and another desktop system, can configure X.desktop to match the appearance and behaviour of the other system. Thus they can minimise the errors associated with transfer, and reduce the amount of relearning needed.

Associating data files with programs

By defining appropriate icon rules, each data file can be associated with the most relevant program, so that double-clicking on the data file icon invokes the appropriate program with the data file supplied to it.

Dragging a data file on to a program icon can be used as an alternative method of invoking the program. For example, a rule file could be written such that compiler source files were edited by dragging them on to the editor icon, and compiled by double-clicking on them.

Background mail server

The rule files are flexible enough to allow the creation of applications such as a mail server which posts new mail as an icon on the desktop. The mail server would run as a background task, and when mail was received it would run an X.desktop command language command to place the mail file on the desktop. The rule file could specify that double-clicking on a mail file icon would open it in a text editor and delete the original file.

Waste icon

The Waste icon is created in X.desktop using rule files, with no additional programming. The Waste icon is a directory, with an appropriate title and picture. It contains a rule file that causes any icon either dropped into the directory window or on to the directory icon to be moved to the directory. The Waste directory can be cleared by double-clicking on its icon with the rightmost mouse button.

The Waste icon would typically be locked on to the desktop by including it in the locked files list.

Automatic encryption/decryption

The drop rules allow directories with special characteristics to be created within X.desktop. For example, a directory could be created such that all files dragged into it are encrypted. They could be decrypted either by dragging them into another directory window, or by double-clicking on their icon.

Local rule files

Rule files can be local to one user, or even local to a specific directory. User-specific rule files can take care of the different computers that different users on a UNIX network might be using. Each user can thus double-click a program file on the network, and run it on a computer appropriate to that program.

Changing environments

The X.desktop system is flexible enough to allow a single user to switch environments at will, thus changing both the appearance and behaviour of the desktop, by double-clicking the appropriate environment file.

For example, a user might have two characteristic modes of working, programming and documenting. In the first mode the desktop might display compilers and debuggers, and the rule files could specify that double-clicking source files would run the appropriate compiler. In the documenting mode, the desktop might display word processing and flow-charting programs, and double-clicking source files would load them into an appropriate word processor.

The standard rules take any file ending in `.xde` as an environment file. Double-clicking an environment file with the lefthand mouse button changes the desktop to that environment.

Message files

The message file contains all the messages used by X.desktop in a standard editable form. You can alter this file to tailor the messages to specific applications: for example, more explanatory messages in a teaching environment or terse messages in a development environment. Or you can change the file to cater for different languages.

Section 2
Tutorials

This section consists of four worked examples, based on configuring the behaviour of X.desktop to facilitate the use of a file compression utility. These examples illustrate most of the main features of the X.desktop rule files, and should serve as a useful basis for creating other applications.

Determining the appearance of your desktop

By creating rule files you can assign icons to specific files or directories, and you can specify a title for the icon to be displayed instead of the usual file title. The following example illustrates the use of rule files by defining a title and icon for the UNIX file compression utility, `compress`. Make a copy of the UNIX `compress` program in a suitable directory in your user file space.

Creating the icon

First we need to define a suitable icon, and put it into the picture file directory `/usr/x/lib/xdtpictures`.

The simplest procedure is to start with an existing picture file, as follows:

1 copy an existing picture file, or the blank picture file `blank.px`, and rename it `compress.px`

2 double-click its icon; this will run the bitmap editor, and you can then edit the icon with the mouse pointer

3 exit from the bitmap editor by clicking **Quit** when you are satisfied with the picture.

In the bitmap editor the mouse buttons have the following functions:

Left	set pixel black
Middle	change pixel
Right	set pixel white.

Refer to Volume 3 of the *X Window System User's Guide* for full details on using the bitmap editor.

Defining an icon rule

The next step is to define an icon rule assigning the appropriate picture and title to the compress utility. We will give the utility the title `Squash`.

Icon rules have the format:

```
ic { [ file-spec { file-rules } ] … }
```

where `ic` is a label identifying icon rules, and *`file-spec`* is a construct specifying which icons the rules should apply to. In this example only the file `compress` is to be affected.

`file-rules` gives a list of rules specifying the appearance and behaviour of all the selected icons. In this example the rules are:

```
ti =Squash; pi = compress.px
```

The label `ti` introduces a title for the icon; in this case `Squash`.

The label `pi` introduces the picture to be displayed for the icon instead of its default picture; in this case, the one we have defined in the bitmap editor.

The full icon rule is thus:

```
ic { compress { ti =Squash; pi = compress.px } }
```

Adding the rule to a rule file

Put this rule in a file named `.xdtdirinfo` in the directory containing `compress`, and the icon will change to display the picture and title you have designed.

If this rule file already exists you will need to incorporate the new icon rule into the rules it already contains. If there are no other icon rules in the file starting with `ic`, you can put this rule at the end of the file as it stands. Otherwise you will need to insert this rule at the end of the list of existing icon rules, so that they read:

```
ic {
      existing-icon-rules ;
      compress { ti = Squash; pi = compress.px }
   }
```

Affecting a group of icons

It might perhaps be useful if files compressed with the `compress` utility are given the same icon as the utility itself – or a related one if you prefer to design a new icon.

We can do this very simply by making use of the facility to define groups of files in the rule files.

The UNIX compress utility gives the compressed versions of files the suffix `.Z` after their filename. The rule file can assign an icon to all such files by giving the file specification:

```
*.Z
```

where `*` is a wildcard matching any filename.

The icons affected can be restricted to just files or just directories by the suffix `/F` or `/D` respectively, so it would be better practice to give the file specification as :

```
*.Z / F
```

The full rule then becomes:

```
ic {
       *.Z / F { pi = compress.px }
    }
```

Files called *`filename`*`.Z` will now automatically display the appropriate icon to identify them.

Building intelligence into your file icons

The rule files also allow you to specify an action to be carried out when an icon, or group of icons, is triggered. Usually triggering means double-clicking with one of the mouse buttons.

In this example, we will define a rule that will automatically uncompress a file that is in compressed format if it is double-clicked with the lefthand mouse button.

We will assume that all compressed files have names with the suffix `.Z` as described above, and that the program `uncompress` is available.

By a simple extension of this example, you could configure your system so that every text document or data file invokes the appropriate program tool when double-clicked, according to its filename.

Using mouse triggers

The standard mouse triggers are double-clicks with one of the mouse buttons, and to make X.desktop as portable as possible these triggers are normally predefined with names as follows:

`s1`	double-click on mouse button 1, the leftmost button
`s2`	double-click on mouse button 2
`s3`	double-click on mouse button 3.

The definitions of `s1`, `s2` and `s3` are given in the X.desktop defaults file, and can be altered to provide alternative ways of producing the three triggers on systems with fewer than three mouse buttons.

Writing trigger rules

The action to be performed when an icon is double-clicked is defined in the icon rules by a trigger-action rule:

```
ta: trigger-id { action-list }
```

where the label `ta` identifies the clause as a trigger action.

The *`trigger-id`* specifies one of the predefined mouse triggers, as set in the defaults file. Here we will use `s1`, a double-click with the leftmost mouse button.

The *action-list* specifies the commands that are actually run when the specified *trigger-id* is applied to an icon in the specified *file-spec*.

Each command in the *action-list* has a prefix indicating whether it should be run by the standard UNIX shell, or by X.desktop.

Here the action is to run the program `uncompress` with the file as parameter, to create a new file of the same name, but without the `.Z` suffix.

The full rule is:

```
ic {
      *.Z/F { ta : sl
               {  ac
                  {
                     b : uncompress <%P0 >%D0/`basename
                                             %B0 .Z` ;
                     d : ddw %D0
                  }
               }
            }
   }
```

This rule illustrates substitutions, which can be used in rules to refer to the components of the names of the files they apply to.

The following substitutions are used:

`%P0`	the absolute pathname of the file
`%B0`	the basename of the file
`%D0`	the dirname of the file.

The first command in the action list uncompresses the double-clicked file to a file with the same name but without a `.Z` suffix. It constructs this name out of the file's dirname and basename, using the UNIX `basename` command to return a filename with the `.Z` stripped off.

The second command in the action list runs the X.desktop command `ddw` to redraw the directory window to show the changed file icons.

This rule can be combined with the rule in the previous tutorial defining the appearance of `.Z` files. It should be included in the file `.xdtdirinfo` in the directory containing the compressed files. These files will then automatically run the `uncompress` utility, and create an uncompressed version of the file in the same directory, when they are double-clicked with the lefthand mouse button.

Loading files into a program by dragging

Another convenient way of telling X.desktop to run a program with a file as data is to drag the data file's icon on to the program's icon.

This action can also be specified in the rule files by making use of the drag triggers `d1`, `d2` and `d3`. These triggers are sent to an icon when another icon is dragged on to it, and dropped, with the corresponding mouse button.

Since potentially any type of file could be compressed, it is inappropriate to use double-clicking to activate the `compress` utility. However, for this task we can make use of the dynamic triggers. In the next example we will define a rule so that any file dragged on to the compress program with the leftmost mouse button is automatically compressed to create a `.Z` file.

The full rule is:

```
ic {
      compress/FX { ta : d1
                     {ac
                        {
                           b : %P0 <%P1 >%P1.Z ;
                           d : chk %P1.Z
                        }
                     }
                  }
   }
```

Here the pathname of the dragged file is substituted for `%P1`, and the first command in the action list runs compress (`%P0`) on this to create a file of the same name with a `.Z` suffix. The X.desktop command `chk` then updates the new icon.

This rule should be included in the `.xdtdirinfo` file of the directory containing the `compress` program.

Building intelligence into directories

It is often useful to organise files into directories on a functional basis, such that all files of one type are kept together in one directory. The rule files conveniently allow actions to be performed when files are dragged into a directory window, making it possible to give certain directories in the filing system special attributes.

In the next part of the tutorial, we will create a special directory called `compact`. Files dragged into this directory will automatically be compressed, and the original version of the file will be deleted to save the user's file space.

Drop rules

The action of dropping an icon into a directory is defined by the drop rules. These have a similar form to the rules defining the action of double-clicking icons, except that there is no file specification. To create a drop rule specific to one directory, you have to put the rule file, named `.xdtdirinfo`, into the directory. Drop rules in user or system rule files apply to all directories.

The drop rule has the format:

```
dd { [ td : dynamic-trigger-id { action-list } ] … }
```

where the *`dynamic-trigger-id`* specifies one of the predefined mouse triggers. Here we will use `d1`, a drag using the lefthand mouse button.

As before, the *`action-list`* specifies the commands to be carried out when the icon is dropped into the directory window. In this case, the action is to compress the file dropped, whose pathname is given by `%P1`. Then we wish to move the resulting compressed file into the directory and delete the original file:

```
dd {
        td : d1   {
                        d : mvi %P0 %P1 ;
                        b : compress %P0/%B1 ;
                        d : ddw %P0
                  }
   }
```

In this rule, we make use of the fact that %P1 contains the file's pathname, and %P0 the pathname of the directory into which it was dropped.

The first command in the action list moves the file into the directory, and redisplays its icon there. The second command compresses it, using the directory pathname %P0 to refer to the file in its new location, and finally the ddw command redraws the directory window to show the file's new icon.

This rule should be included in the .xdtdirinfo file in directory compact, which is to have this special action.

An extension

A final refinement is to have the same action take place if we simply drag a file icon on to the icon of directory compact, rather than into its directory window. We can achieve this by including an additional icon rule in the rule file:

```
ic {
      compact/D    { ta : dl
                        { ac
                           {
                              d : mvi %P0 %P1 ;
                              b : compress %P0/%B1 ;
                              d : ddw %P0
                           }
                        }
                     }
   }
```

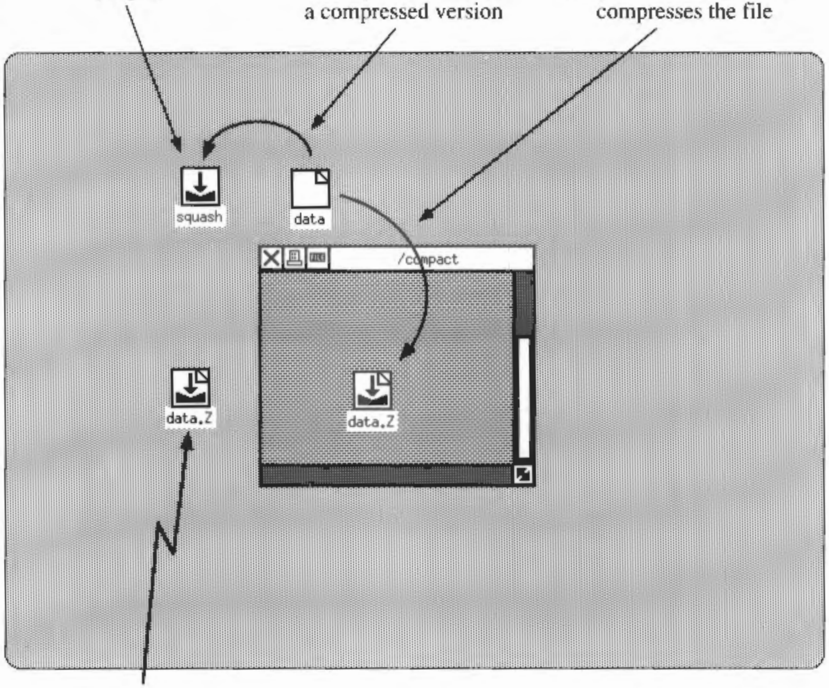
The complete system

Fig. 3 shows the combined effects of the rules we have defined in the preceding tutorials. The rules could all be combined into two rule files: a file .xdtdirinfo in directory compact, containing the drop rule for that directory, and a file .xdtuserinfo, containing all the other rules.

File compaction program

Dragging a data file on to the squash icon makes a compressed version

Dragging an icon into the 'compact' directory window compresses the file



Double-clicking the compressed version (with a .Z suffix) uncompresses it

Fig. 3: Behaviour of the example rules in tutorial.

Section 3
Reference

Rule files

Introduction

Rule files are used to specify the behaviour and appearance of the icons representing files in X.desktop. For example, they enable you to associate a picture and special mouse click actions with specific files or groups of files and to define what to do when one icon is dropped on to another.

Rule files are text files, and can be edited with a normal text editor. Special configuration tools may also be available to help with this task.

Components of rule files

A rule file consists of a number of components which may occur in any order, but each component should only occur once in any one rule file.

The different components in a rule file are:

Icon rules – describe what the icon for each file looks like – which picture file is used to display it, and what its title should be – and what should happen when the icon is triggered, by double-clicking it or dragging another icon on to it.

Drop rules – describe what happens when icons are picked up and dropped on to the background of a directory window.

Desktop layout – lists which icons are out on the desktop, and their positions. It is normally generated automatically.

Locked files – these are files which are locked out on the desktop and cannot be put back.

The rule hierarchy

There are four kinds of rule files, and they generally have the following precedence:

Local rule files – can be found in any directory, and provide rules that apply only to the files in that directory. They all have the name `.xdtdirinfo`.

Environment rule files – these are sets of rules and a list of files that are out on the desktop. Environments allow users to have different rules for different circumstances; for example, a user might have a programming environment and a text editing environment. At any time, each user has one active environment rule file, known as the 'current environment'. When a user changes environment, the icons currently on the desktop are saved in the old environment rule file and put away, after which a new set is read from the new environment rule file and placed on the desktop.

Environment rule files can have any name, though it is suggested that they should end in `.xde`. The initial environment rule file for a user is specified through the X defaults mechanism.

User rule files – each user can have a user rule file called `.xdtuserinfo` stored in the user's 'home' directory. It applies only to that user.

System rule file – applies to all desktops running on a given machine. There is only one such file: `/usr/x/lib/xdtsysinfo.`

System and user rule files are preloaded; therefore any changes made to these will only take effect when X.desktop is next run.

Writing rule files

Rule files are pure text files, and so can be created and edited using any suitable program editor such as `vi` or `xedit`. In general, the layout is not critical, and spaces or new lines can be inserted to improve readability, and make the structure of the rules clearer.

For example, the following two rule files are equivalent:

```
ic{*/d{pi=dir.px;}*/f{pi=file.px;}}
```

and:

```
ic {
      */D    {
                 pi = dir .px ;
             }
      */F    {
                 pi = file.px ;
             }
   }
```

In general, spaces and new lines should be used to clarify the layout and function of rule files.

Special characters

The following four characters have special meanings in rule files:

Percent	`%`
Open brace	`{`
Close brace	`}`
Semicolon	`;`

Braces are used to group together similar items, semicolon is used to terminate other items, and percent introduces special phrases and instructions (the character percent was chosen to be distinct from the characters used for this purpose in UNIX).

When a semicolon would be followed by a close brace it may be omitted.

There are a few places, such as where an icon title is specified, where spaces are significant. For example, the spaces in the following rule are not ignored:

```
ic { * { ti =Title for all icons; }}
```

If necessary, new lines and spaces can be included for formatting purposes in the icon title by surrounding them with a pair of % signs, and they will then be ignored. So the following rule is identical to the previous example:

```
ic { * { ti =Title %
         %for all icons; }}
```

Escape sequences

The characters % { } ; can be included in rule files (such as in the title of a filename) by preceding them with a % sign.

Other characters can be included by preceding them with the escape sequence:

```
%!
```

Control characters

You can include a control code, or other special character, in a rule file using the following sequence:

```
%# decimal-code #
%#0 octal-code #
%#0x (or %#0X) hexadecimal-code #
```

For example, the character double quote, character code 34, can be written as:

```
%#34#
%#042#     (34 decimal = 42 octal)
%#0x22#    (34 decimal = 22 hexadecimal)
%#0X22#
```

Comments

Comments can be included in a rule file by prefixing them with the sequence %// which causes all characters up to the end of the line to be ignored. Long comments can be preceded by the sequence %/* and followed by the sequence %*/ which causes all characters between these two to be ignored.

Note that comments introduced with the sequence %/*, should obviously not contain the character sequence %*/, as this would indicate the end of the comment.

Referring to filenames

When a file is referred to, its name may be used in four ways. These four ways have special (UNIX) names:

Absolute pathname – this is the full name of the file, and always begins with a slash.

Basename – this is the name of the file within its directory. It is the part of the absolute pathname following the last slash.

Dirname – this is the name of the directory holding the file. It is the part of the absolute pathname preceding the last slash.

Relative pathname – if the file is in the desktop working directory, or one of its subdirectories (ie if the absolute pathname begins with the absolute pathname of the working directory, followed by a slash), then it is the absolute pathname with the name of the working directory, and the following slash, removed. Otherwise it is the same as the absolute pathname.

For example, the various names of the file `/user/fred/work/letter` are:

Absolute pathname:	`/user/fred/work/letter`
Basename:	`letter`
Dirname:	`/user/fred/work`

The relative pathname depends on the working directory:

Working directory	**Relative pathname**
`/`	`user/fred/work/letter`
`/user`	`fred/work/letter`
`/user/fred`	`work/letter`
`/user/fred/work`	`letter`
any other	`/user/fred/work/letter`

Note that there is one special case. The dirname of `/` is `/.` (slash-dot) and its basename is `/` (slash).

Components of rule files

This section describes the main components common to both icon rules and drop rules.

Triggers

To give X.desktop the flexibility to work with any type of mouse, with from one to five buttons, the rules are usually expressed in terms of triggers rather than physical buttons.

Each trigger is labelled with a trigger-id, which is the letter s or d followed by a number. Each trigger-id is assigned to a particular sequence of clicks, or presses, of the mouse buttons according to the X defaults file.

The letter s is used for 'static' triggers, where the mouse is not moved (eg double clicks), while d is used for 'dynamic' triggers, or drags (eg dropping one icon on another).

For example, the usual assignment for a three button mouse is for the three trigger-ids s1, s2, and s3 to be double-left, double-centre, and double-right click. Another user, with only two buttons on the mouse, could set them to be double-left, double-right, and double-both.

Most users will not change the trigger to trigger-id mapping, since this will have been optimised to the particular mouse supplied with the computer system.

Action lists

Action lists specify the commands that X.desktop runs when an icon or directory window is triggered.

Syntax: {ac { *control* : *command* } }

An action list can be empty, or contain one or more commands separated by semi-colons. Each command is prefixed by a control letter, specifying how the command is to be run:

b the command should be executed by the standard UNIX shell /bin/sh

t the command should be executed by the standard UNIX shell within the standard terminal emulator (normally xterm)

d the command should be executed by X.desktop; see *Desktop command language*.

Example:

```
ac {
      t  :  vi myfile.1                   ;
      b  :  troff <myfile.1 >myfile.1.trf ;
      d  :  chk myfile.1.trf
   }
```

The commands are executed in order, with each being carried out after the previous one has finished. However, while X.desktop is waiting for one command to finish executing, it may be doing other things, and several action lists may be executing at the same time.

Substitutions

Substitutions allow rules to make use of the filenames that correspond to the icons triggered, or dropped.

The name of a file may be substituted into the title of its icon and the names of any of the files involved may be substituted into commands in action lists.

Substitutions are made by placing a substitution sequence in the rule file at the appropriate point. A sequence consists of a percent sign, a letter indicating the information to be substituted, and a number or asterisk.

The number or asterisk determines the file or files for which information is to be substituted. The permitted cases are:

In icon titles:
`0` the file of the icon

In action lists for static icon triggers:
`0` the file of the icon

In action lists for dynamic icon triggers:
`0` the file of the icon dropped on to

`1-9` the appropriate file in the list of those dropped on to the icon

`*` the files of each of the icons dropped in turn – the values are separated by spaces. For example, if three icons are dropped, then `%P*` is the same as `%P1 %P2 %P3`.

In action lists for drop rules:
`0` the directory of the directory window dropped into

`1` the file of one of the icons dropped into the window; see *Drop rules*.

The letter may be any of the following:
`P` the absolute pathname of the file

`B` the basename of the file

`D` the dirname of the file

`R` the relative pathname of the file, except in the titles of icons within directory windows, when it is the basename of the file

`C` the class of the file (four capital letters); *see Classes*

`N` when followed by an asterisk gives the number of files dropped; for example, if three icons are dropped `%N*` is 3.

For example, suppose that the local rule file for the directory `/fred/jim` is:

```
ic
{
   jane  { ta : s1 { ac {  b : %R0 -z       } } }
   sarah { ta : d1 { ac {
                           b : mv %P* %D0;
                           b : echo %N* >%D2/count
         }         }     }
}

dd
{
   td : d1 { d : lni %P0/new %R1 }
}
```

and suppose that the current working directory is `/fred`, then the following action lists will be generated by the indicated actions:

Trigger-id `s1` on `/fred/jim/jane` :

```
b : jim/jane -z
```

When several icons are dropped into an icon, the trigger-action rule is executed once with the full list of files dropped. For example, drop the icons `/fred/one` , `/jim/two`, and `/ian/my/data` on to `/fred/jim/sarah` with trigger-id `d1`:

```
b : mv /fred/one /jim/two /ian/my/data /fred/jim ;
b : echo 3 >/jim/count
```

When several icons are dropped on to a directory window, the drop rule is executed once for each file dropped. For example, drop the same icons into the directory window for `/fred/jim` with trigger-id `d1`:

```
d : lni /fred/jim/new one
d : lni /fred/jim/new /jim/two
d : lni /fred/jim/new /ian/my/data
```

Icon rules

Icon rules describe the behaviour and appearance of files when represented by icons:

- the picture file used for the icon
- the text for the icon's title
- the action when the icon is triggered (double-clicked)
- the action when another icon is dragged on to the icon.

Syntax: `ic { [ file-spec { file-rules } ] … }`

The *`file-spec`* specifies a file, or group of files, to which the *`file-rules`* apply. It is usually clearer to group all the rules that apply to one class of files together.

The file specification

The file specification restricts the files to which a clause applies – the 'ruled files'.

Syntax: *`filename`*

or: *`filename / class`*

where *`filename`* is an ambiguous name; *see* below.

If no class is specified the rule applies to all classes.

Ambiguous names

Ambiguous names are filenames, optionally containing certain special characters or wildcards. The following characters can be used in ambiguous names:

`?` any character. For example, `a?c` includes the files `abc` and `aac` , but not the file `abbc`.

`*` any sequence of characters, including none.

For example, `a*c` includes the files `ac` , `abc` , `acbc` , and `as4hx..o6f,s:c`.

`[abc]` any of the specified set of characters.

The set of characters can be abbreviated using a dash (minus sign) to represent a range; for example, `A-D` is equivalent to `ABCD`. Ranges should only be between two letters of the same case, or two digits.

Prefixing the set with `^` (circumflex) means not any character specified. For example, `[^A-Za-Z]*` means any file beginning with a character other than a letter.

Note: The special filename / may appear in the ruled files by writing it as `//d` ('files called slash which are directories').

Classes

Classes are used to represent the properties of files in a concise form. These properties fall into four sets, and a file has exactly one property of each set. The properties are each represented by a letter (of either case), so that the class of a file consists of exactly four letters.

The letters used are as follows:

File type

`B` block special file

`C` character special file

`D` directory

`F` regular file

`G` ghost (non-existent) file

`I` inaccessible file

`P` pipe

`S` symbolic link to non-existent file; on machines without symbolic links no files have this type.

Execute permissions

`X` the user may execute the file

`A` the file has execute permission for someone, but not for the user

`N` the file does not have execute permission for anyone.

Read/write permissions

`W` the user may read and write the file

`V` the user may read but not write the file, and the file has write permission for someone

`K` the user may read the file, and the file does not have write permission for anyone

`H` the user may not read the file.

Ownership

`M` the user owns the file

`O` the user does not own the file.

Note: The codes `G`, `I`, and `S` imply the codes `H`, `N`, and `O`.

The term 'class' in general refers to a set of options from the above table, and is described by a string of the appropriate letters.

The order of the letters is not significant, and redundant letters are ignored.

For each set of properties, the letters of that set that appear should be viewed as being separated by the word 'or', with the groups of letters from different sets being separated by 'and'.

For example, class `BCNWVO` is '(`B` or `C`) and `N` and (`W` or `V`) and `O`'.

If no letters of a class appear, then the class is read as if they all appeared. For example, class `D` is the same as `DXANWVKHMO` (both mean 'all directories').

A number of extra codes may also be used in classes. These each stand for a common combination of the standard codes:

`Q`	`G`, `I`, or `S`	(not a real file)
`E`	`X` or `A`	(executable by somebody)
`U`	`A` or `N`	(not executable by the user)
`L`	`V` or `K`	(readable but not writable by the user)
`R`	`W` or `L`	(readable by the user)

For example, `DEO` and `DAXO` have the same meaning.

The following classes are the most useful in rule files:

`D`	directories
`F`	files
`FE`	executable files
`FX`	files executable by the owner
`FN`	data files
`FNW`	data files that the owner can alter
`FNR`	data files that the owner can read

Picture specification

Assigns a picture file to the files specified in the file specification.

Syntax: `pi = ` *`picture-file`*

For example:

```
* / D { pi = dir.px }
```

means that all directories are to use the picture in the picture file `dir.px`.

Title specification

The title specification assigns a title to the specified files.

Syntax: `ti = ` *`title`*

All spaces are significant between the equals sign and the semicolon or closing bracket. For example, the following clause sets the title of the `xcalc` program:

```
xcalc { ti =Calculator}
```

When using an ambiguous name, substitutions may be used to include the actual filename in the title.

Trigger actions

Trigger action rules are used to specify an action to be carried out when a specific trigger occurs with the mouse pointing to the icon.

Syntax: `ta : ` *`trigger-id`* `{ ` *`action-list`* ` }`

The `action-list` specifies a list of commands to be carried out by X.desktop or the operating system of the computer.

For example, the following trigger rules say that trigger-id `s1` on the appropriate icon causes the `xclock` program to be run:

```
ta : s1 { ac { b : xclock }}
```

Alternatively, the action list can be blank, in which case the trigger-id will have no effect. For example, the following clause says that trigger `s3` on a directory should do nothing:

```
* / D { ta : s3 { } }
```

The following more complex example illustrates the use of trigger actions:

```
ic {
      *.c       { pi = csrc.px; }
      * / D     {
                   pi = dir.px;
                   ta : s1 { ac { d : ddw %P0 t }}
                }
      [ab]*     { ti =AB file %B0; }
   }
```

This has the following effect:

- all files whose names end in `.c` use the picture found in the picture file `csrc.px`
- all directories use the picture found in the picture file `dir.px`
- when any directory is triggered with the trigger `s1`, then a directory window is opened to show that directory in time order
- any file whose name begins with the lower case letters `a` or `b` has the icon title `AB file` followed by the basename of the file.

The rules take effect in the order in which they are specified, so the first rule in this example will only affect files which have not already been given a picture by a previous rule.

Likewise, the second clause (pictures for directories) does not apply to directories whose name ends with `.c` (though the third and fourth ones do).

Examples of icon rules

The following example defines icons and titles for the calculator, clock, and editor programs, and causes the editor to be run when a file icon is dragged on to its icon, with that file loaded ready for editing:

```
ic
   {
      xcalc    {  ti =Calculator; pi = xcalc.px  }
      xclock   {  ti =Clock;      pi = xclock.px }
      xedit    {  ti =XEdit;      pi = quill.px;
                  ta : s1 {}  ta : s2 {}  ta : d2 {}
                  ta : d1 { ac { b : %P0 %P1 }}
               }
   }
```

The next example defines a rule which moves any file or files dropped on to a directory icon with the lefthand mouse button into that directory:

```
ic
   {
      */D   {
               ta : d1{ ac {d : mvi %P0 %P*}}
            }
   }
```

The final example shows the standard definition of the icon rules for the Waste icon. The Waste icon is implemented as a directory with a suitable picture and title.

For convenience, dragging with the leftmost mouse button moves a file to the Waste directory, rather than copying it as is the usual default.

The Waste directory can be emptied by double-clicking with the third mouse button. The command `rm -rf %P0/*` deletes all files in the directory.

```
ic
{
   waste /d
   {
      ti =Waste;
      pi =waste.px;
      ta : d1 { ac { d : mvi %P0 %P* } }
      ta : d2 { ac { d : mvi %P0 %P* } }
      ta : s3 { ac { b : rm -rf %P0/* } }
   }
```

Drop rules

Drop rules describe the effects of dropping icons into directory windows.

Syntax: `dd {[ td :` *`dynamic-trigger-id`* `{` *`action-list`* `} ]... }`

Drop rules in local rule files apply to the directory window of the directory holding the rule file. Drop rules in other rule files apply to all directories.

Drop rules consist of a set of action lists, each associated with a dynamic trigger-id. As with icon rules, the first match is used.

For example, the following drop rules cause dragging an icon into a directory window to copy or move the file, depending on whether mouse button 1 or 2 is used. This is usually the default behaviour for X.desktop. In each case `%P1` is replaced by the pathname of the file dropped, and `%P0` by the pathname of the directory window into which it was dropped.

```
dd
{
   td : d1 { d : cpi %P0 %P1 }
   td : d2 { d : mvi %P0 %P1 }
}
```

Multiple drops

An important difference to note between drop rules and icon triggers is the action taken when several icons are dropped into a directory window at the same time. The action list is duplicated several times, one copy for each icon dropped, and then each copy is modified by the substitution system to include details about that icon.

For example, suppose we have the action list:

```
{
   d : mvi /waste %P1                ;
   b : echo %P1 >>/waste/.filelist
}
```

and we drop the three icons `/fred/fred`, `/fred/jim`, and `/fred/sheila`. Because `%P1` is replaced by the pathname of the icon dropped, the action list actually executed is:

```
{
   %// Spaces have been added to the commands in
   %//  order to make their meaning clearer.
   d : mvi  /waste /fred/fred     ;
   b : echo /fred/fred   >>/waste/.filelist ;
   d : mvi  /waste /fred/jim      ;
   b : echo /fred/jim    >>/waste/.filelist ;
   d : mvi  /waste /fred/sheila   ;
   b : echo /fred/sheila >>/waste/.filelist
}
```

Desktop layout

The desktop layout list describes the files which are on the desktop, together with their positions. It is normally generated automatically by X.desktop, but could be modified by a program to alter the initial appearance and layout of a desktop. The desktop layout list is ignored if it is not in an environment file.

Syntax: `dt { [ filename [@ position ] ; ] … }`

The *`position`*, if present, consists of one of the following:

`G` followed by the coordinates in tidying grid units

`P` followed by the coordinates in pixels

`F` indicating that the icon is to be placed at the first free position of the grid.

Omitting the position code is the same as specifying a code of `F`.

For example:

```
dt {
      /            @  G   0,    0 ;
      /usr/bin     @  G   1,    0 ;
      /fred        @  F           ;
      /fred/main   @  G   4,    7 ;
      /bin         @  P 211,  874 ;
      /fred/data
   }
```

Locked files list

The locked files list allows icons to be locked on the desktop.

Syntax: `lf { [ filename ; ] … }`

Locked files lists in the system and user rule files apply whenever X.desktop is run. A locked files list in an environment file applies only whilst that environment file is current. Locked files lists are ignored in local rule files.

For example:

```
lf {
       /          ;
       /bin       ;
       /usr/bin
    }
```

Mapping triggers

In order to be more easily portable, X.desktop converts clicks on the mouse buttons first into triggers, and then into trigger-ids. This mechanism is controlled by four items in the X defaults mechanism; these are the mapping (a string), the maximum motion (a number of pixels), a threshold down time, and maximum up time (both times measured in milliseconds).

The trigger mapping set up when your system is supplied will normally be optimum for your mouse and configuration, but you can modify the actions to suit your own requirements.

The conversion is done in two stages. Firstly, the motions and button presses are converted into triggers. This is controlled by the three latter items. Secondly, the triggers are then converted to trigger-ids through the mapping string.

Triggers

A trigger is a set of closely spaced button presses and releases. The easiest way to think of a trigger is as a series of 'steps'. Each step starts when, with all the mouse buttons up, one of the buttons is pressed. It ends the next time that all the buttons are up.

Trigger steps

A step is labelled by giving the numbers of all the mouse buttons that are depressed at any time during the step, no matter what order they are in, or how long they are down.

For example, all three of the following examples would be labelled as `1` and `3`, or `13` for short:

Press button 1, press button 3, release button 1, and release button 3.

Press button 3, press button 1, release button 1, and release button 3.

Press button 1, press button 3, release button 3, press button 3, release button 3, and release button 1.

Note: 'press' means press and hold down the button.

The three types of step are defined as follows:

Step	Mouse movement	Interval between events
short click	< maximum motion	< threshold down time
long click	< maximum motion	> threshold down time
drag	> maximum motion	—

A short click is described by giving the numbers of the mouse buttons: `13`

A long click is described by giving the numbers of the mouse buttons followed by a minus sign: `13-`

Triggers

A trigger is a sequence of steps, and is described by giving the steps, separated by commas. For example, the trigger 'double-click on button 2' would be described as `2,2`.

If the last step in the sequence includes a drag, the trigger is defined as a dynamic trigger, and X.desktop will signify detection of the drag by changing the cursor to the drag or multi-drag cursor. Other triggers are defined as static triggers.

A trigger ends when either there is no further event for the maximum up time after a step, or at the end of a drag, whichever comes first.

All triggers containing more than five steps are ignored by X.desktop.

Converting triggers to trigger-ids

All triggers which you want to be interpreted by X.desktop must appear in the mapping string. This consists of a sequence of mappings, separated by semicolons (spaces anywhere in the mapping string are ignored).

There are three things that can occur in the mapping string:

- static trigger mappings
- dynamic trigger mappings
- macro definitions.

Static trigger mappings

Each static trigger mapping maps a static trigger to a trigger-id.

Syntax: *`static-trigger = trigger-id`*

where *`static-trigger`* is a list of steps, separated by commas.

`trigger-id` is `s` followed by a number.

A static trigger may also be used to control the selection of icons. This is done by using one of the following codes instead of a trigger-id:

`+s` if the current icon (the icon which is under the cursor) is not selected, then select it

`-s` if the current icon is selected, then deselect it

`!s` deselect all selected icons, and then select the current icon

`~s` if the current icon is selected, then deselect it. Otherwise, select it.

For example, the following says that a short click on button 1 selects the current icon in addition to any icons already selected, whilst a long click selects it on its own. Either type of click on button 2 deselects the icon:

```
1=+s ;   1+=!s ;  2=-s ;   2+=-s
```

Dynamic trigger mappings

Each dynamic trigger mapping maps a dynamic trigger to a dynamic trigger-id.

Syntax: *`dynamic-trigger = trigger-id`*

where *`dynamic-trigger`* is a list of steps separated by commas.

`trigger-id` is `d` followed by a number.

For example, the following says that a drag with button 2 on its own generates trigger-id `d2`, but if preceded by a short click on button 4, it generates trigger-id `d6`:

```
2=d2 ; 4,2=d6
```

Dynamic triggers cannot be used to control icon selection.

Macro definitions

Macro definitions allow one or more buttons to be abbreviated to a single letter. This allows mappings to be made more abstract, and so easier to convert for a different number of buttons.

For example, suppose that you have designed a set of mappings for a three button mouse, and that you want to convert it to work on a two button mouse. One way might be to say that the centre button is represented by using both left and right buttons together. By specifying all the mappings in terms of the letters L, C, and R, rather than the numbers 1, 2, and 3, they are easier to change (especially as the right button will change from being number 3 to being number 2).

A macro definition consists of a set of button numbers, an equals sign, and then a single letter. That letter can then be used in any future trigger description or macro definition.

The above example, with three static trigger-ids, three dynamic trigger-ids, and three selection control triggers, might be written as follows for the three button mouse:

```
1=L      ; 2=C       ; 3=R  ;      \
L=!s     ; C=+s      ; R=-s ;      \
L,L=s1   ; C,C=s2    ; R,R=s3      ;     \
L=d1     ; C=d2      ; R=d3
```

(the backslashes indicate that the mapping is continued on the next line).

To convert to the two button mouse, we change the first line to:

```
1=L      ; 12=C      ; 2=R  ;      \
```

The mapping then becomes equivalent to:

```
1=!s     ; 12=+s     ; 2=-s ;      \
1,1=s1   ; 12,12=s2  ; 2,2=s3      ;     \
1=d1     ; 12=d2     ; 2=d3
```

Desktop command language

The X.desktop command language, XCL, allows certain actions to be carried out within X.desktop. These actions are mainly concerned with icons, directory windows, and copying and moving files.

X.desktop command language functions can be piped into X.desktop or used in rule files.

The advantages of using XCL commands are that they automatically take care of updating the desktop, and they do not rely on the availability of particular UNIX binary files.

Commands in XCL consist of a number of words, separated by spaces. The end of a command is determined by the mechanism which initially generates the command – for example, within a rule file, the end of a command is indicated by a semicolon. A backslash causes the following character to be part of the current word, even if it is a space character. For example, the following command contains only two words:

```
this\ is\ a\ command   with\ only\ two\ words
```

All valid commands begin with a word of three lower-case letters, followed in some cases by a number or arguments. Some commands require an exact number of arguments, and the effect of having the wrong number is undefined. Other commands will accept any number of arguments.

The following notation is used for describing the arguments of commands:

`file`	represents a filename
`dir`	represents an existing directory.

Desktop commands

Terminate execution – `die`

Syntax: `die`

Terminates X.desktop execution.

Change environment – `ndt`

Syntax: `ndt` *`file`*

Changes the current current X.desktop environment to the specified file.

Catalogue desktop – `cdt`

Syntax: `cdt` *`file`*

Writes a list of the icons on the desktop and their positions into the specified file, replacing any such list already in that file. X.desktop's current environment is not changed.

Trigger action – `act`

Syntax: `act` *`static-trigger-id file`*

Executes the action list that would have been executed if the specified trigger had been used on the specified file.

Note that in each case any commands following the `act` command in the action list will be executed immediately, independently of the triggered action list.

Syntax: `act` *`dynamic-trigger-id`* `file [` *`file`* `]` ...

The action list that would have been executed if the list of files had been dropped, using the specified trigger, on the first file named, is executed. Note that this command is completed as soon as the first command in the list *starts* executing.

Drop action – `drp`

Syntax: `drp` *`dynamic-trigger-id`* `dir [ file ]` ...

Executes the action list that would have been executed if the list of files had been dropped, using the specified trigger, on the open window of the directory. The directory window does not need to be open. Note that this command is completed as soon as the first command in the list *starts* executing.

Open directory window – **ddw**

Syntax: `ddw dir [`*`flags`*`]`

Opens the directory window for the directory, if it is not already open, brings it to the front, and displays it in the format given by the flags. Valid flags are:

`i`	display by icon (default)
`n`	display by name;

`a`	sort alphabetically (default)
`t`	sort by time
`c`	sort by class
`u`	sort in 'extra order'.

This command can be used both for opening new windows and for altering the appearance of existing ones.

Replace directory contents – **rdw**

Syntax: `rdw` *`dir1 dir2`* `[`*`flags`*`]`

If the directory window for directory *`dir1`* is open, then its contents are replaced by directory *`dir2`*. The flags have the same meanings as for the `ddw` command. If there is already an open window for directory *`dir2`*, it is brought to the front, and the window for directory *`dir1`* is closed.

Close directory window – **cdw**

Syntax: `cdw` *`dir`*

Closes the specified directory window if it is open.

Bring window to front – **btf**

Syntax: `btf` *`dir`*

Brings the specified directory window to the front if it is open.

Get out icon – `goi`

Syntax: `goi` *`file`* `[`*`position`*`]`

Places the icon of the file on the desktop. If a position is specified, it should be one of the following forms:

P*x*,*y*	position in an exact number of pixels
G*x*,*y*	position in the standard tidying 'grid'
F	first free position on the grid

where *x* and *y* are numbers.

Put back icon – `pbi`

Syntax: `pbi` *`file`* …

Puts back the icons of any of the specified files that are on the desktop (except locked ones).

Tidy desktop – `tdf`

Syntax: `tdf`

Tidies the desktop.

Reorganise desktop – `tds`

Syntax: `tds`

Reorganises the desktop.

Copy file – `cpi`

Syntax: `cpi` *`dir file`* …

Copies the specified files into the specified directory. If the directory window is open, new icons will appear in the window.

Link file – `lni`

Syntax: `lni` *`dir file`* …

Links the files into the specified directory. If the directory window is open, new icons will appear in the window.

Move file – `mvi`

Syntax: `mvi` *`dir file`* …

Moves the files into the specified directory. If the directory window is open, new icons will appear in the window. If the icons of the specified files are on the desktop, their titles will change if necessary. If the icons of the specified files are visible in directory windows, they will disappear.

Update icons – **`chk`**

Syntax: `chk` *`file`* ...

Ensures that any icons visible for the specified files have the correct appearance, even if the properties of the file, or any of the applicable rule files, have changed since the icon was first made visible.

Picture files

The X.desktop icon pictures, background patterns, and control patterns are held in picture files. These can be edited using a bitmap editor.

When the name of a picture file begins with a slash, then the file can be found without help. The picture directory (looked up in the X defaults mechanism) is used by X.desktop to find picture files whose names do not begin with a slash.

If the name of the picture file does not begin with a slash, then it is looked up in two places. Firstly, the name of the picture directory, and a slash, are prefixed to the name of the file. If this file is not found, or if there is no picture directory item in the X defaults, then the standard prefix `/usr/x/lib/xdtpictures` is used instead.

Suppose that the picture directory is set to `/user/fred/pictures` and we are trying to find the picture file `core.pic` . Then X.desktop will look for the files:

```
/user/fred/pictures/core.pic
```

```
/usr/x/lib/xdtpictures/core.pic
```

It is permissible for the picture file to have a slash in its name, so that `patterns/checked.pic` would be looked for in:

```
/user/fred/pictures/patterns/checked.pic
```

```
/usr/x/lib/xdtpictures/patterns/checked.pic
```

Format of picture files

Picture files are an extended form of X bitmap files, and X bitmap files are therefore always legal picture files. Picture files may also be generated with the `pixmap2c` utility (if available on your system).

The format of picture files is being extended by IXI to provide further facilities. All existing picture files will remain valid after any future changes.

A picture file consists of two kinds of items: configuration items and data items. The order of individual items is not constrained except that all configuration items must occur before all data items.

Configuration items

There are six kinds of configuration item.

Each item must be on a separate line, and consists of the prefix `#define` followed by a name and a value, with spaces or tabs separating each of the three parts.

The three parts of the item must all occur on the same line, and the hash sign (#) must be in the first column.

The first part of each item name should correspond to the name of the picture file. In the following examples the items are given for a picture file `pic.px`:

`pic_width`
`pic_height` these two items must occur.

Their values are numbers, and give the width and height of the picture. If the picture is used for an icon, button, or cursor, this will be the size of the object. If it is used as a background, the picture will be tiled across the area; these items are still required to enable the data items to be interpreted.

`pic_x_hot`
`pic_y_hot` these two items must either both occur or both be omitted.

They are only used if the picture is the data portion of a cursor, and indicate the coordinates within the picture where the cursor is actually 'at'. For example, if both values are zero, the actual point of the cursor would be the top left corner of the picture.

`pic_fg`
`pic_bg` these two items must either both occur or both be omitted.

Their values must be names of colours, surrounded by double quotes, giving the foreground and background colours of the picture. If the colour does not begin with a hash sign, then its meaning depends on your X server. If it does begin with a hash sign, then the remainder of the colour name encodes the actual colour.

Your implementation of the X system may interpret spaces in a name. Spaces are not permitted in the encoding format.

The encoding gives the red, green, and blue components of the colour, in that order, as one, two, three, or four hexadecimal digits each (so that components written `5`, `50`, `500`, and `5000` are all the same, and differ from `05`, `050`, and `0005`).

If these items are omitted, then the foreground will be black and the background white.

For example:

black is "#000000000000" or "#000"
white is "#ffffffffffff".

An example of the configuration part of a picture file might be:

```
#define pic_width  10
#define pic_height  8
#define pic_x_hot 5
#define pic_y_hot 4
#define pic_bg "purple"
#define pic_fg "#ffa223"
```

Data items

There is one kind of data item – the picture data. It consists of the sequence

```
static unsigned char pic_bits [] = { data }
```

where `unsigned` may be omitted and *`data`* represents the actual data, consisting of a sequence of two digit hexadecimal values, each prefixed with `0x` and separated by commas.

There may be up to 20 such values per line, though it is usual to have 12.

If the width and height of the picture are W and H respectively, there should be a total of ((W+7)/8)*H values, (W+7)/8 for each row of the picture (the division is rounded down, rather than being an exact number). Each value represents eight consecutive pixels, except that the last value in the row may represent less.

Defaults files

The X defaults mechanism is used by many X utilities to obtain information about which options they are to use. In particular, it is used by X.desktop to obtain a range of information.

The X defaults mechanism works by reading a number of files and constructing a database from them – the full mechanism is described in the X manuals. The database used by X.desktop is built out of the following files (in their order of precedence):

```
$ENVIRONMENT
$HOME/.Xdefaults
/usr/x/lib/xdtdefaults
```

where `$ENVIRONMENT` and `$HOME` are the standard UNIX environment variables. If the file `$ENVIRONMENT` does not exist, then it is replaced by:

`$HOME/.Xdefaults-`*`machine`*

where *`machine`* is the name of the machine that X.desktop is running on.

If this, or either of the other two files, does not exist, it is skipped (so that none of the files are necessary).

If the database does not contain any entries matching a particular item, it uses inbuilt defaults.

Defaults items

Each item is listed in the form:

```
 name         name
         .
class        class
```

Refer to the X windows documentation supplied with your system, or Volume 1 of the *IXI X.Window System Reference Manual* for a full explanation of the defaults system.

Objects that are options should have values which are true or false. The words that are understood by X.desktop are given in the message file, which is explained in Section 3 of this guide.

Text

font
Font

specifies the name of the font that will be used by X.desktop for all text. There is a default font.

```
textMargin
TextMargin                                    (number :2)
```

specifies the amount of space that should appear around all text displayed by X.desktop.

Icon layout

When the reorganise option is used, the icons may be spread out in rows or columns.

```
iconGrid       horizontal
            •
IconGrid       Horizontal                     (true)
```

The default specifies that the icons should be spread out in rows.

```
iconGrid       spacing
            •
IconGrid       Spacing                        (number:100)
```

This specifies the number of pixels apart that icons should be arranged on the desktop when it is tidied. This distance is measured from the centre of each icon.

```
directory      aisleWidth
            •
Directory      AisleWidth                     (number)
```

This is the minimum number of pixels that should be left between each icon in a directory window when it is first opened and whenever it is tidied.

File defaults

```
initialEnvironmentRuleFile
InitialEnvironmentRuleFile      (filename:.xdtinitial.xde)
```

This is the name of the initial environment rule file.

```
isWindowManager
IsWindowManager                                              (true)
```

This option determines whether X.desktop runs as a window manager (true) or as an ordinary program (false).

```
pictureDirectory
PictureDirectory                                 (filename:no default)
```

See Picture files for details on the meaning of this value, which should be the name of a directory and should begin with a slash.

Triggers

The following values are used to convert triggers to trigger-ids.

```
triggers        mapping
            .
Triggers        Mapping                                      (string)
```

The default mapping string (spaced out on several lines) is:

```
1=!s     ; 2=+s   ; 3=-s   ; 4=~s       ;
1,1=s1   ; 2,2=s2 ; 3,3=s3 ; 4,4=s4     ; 5,5=s5 ;
1=d1     ; 2=d2   ; 3=d3   ; 4=d4       ; 5=d5
```

The remaining items are numbers which alter the thresholds used in the conversion. The details of the conversion mechanism are described below.

```
triggers        maxMotion
            .
Triggers        MaxMotion                           (number:3 pixels)

triggers        maxUpTime
            .
Triggers         Time                                 (number:700 ms)

triggers        thresholdDownTime
            .
Triggers            Time                              (number:500 ms)
```

Cursor shapes

Cursors are used by X.desktop for the following functions:

busy — when X.desktop is doing something

drag — when an icon has been picked up and is being moved

multiDrag — when more than one icon has been picked up and is being moved

idle — when X.desktop is waiting for a command menu		when the pointer is over a menu

none		when a window is being moved or resized. This cursor should be blank.

The default cursors are built into X.desktop, but any of them can be redefined. Each pair of pictures forms a cursor shape.

```
 busy      data
        .
Cursor    Bitmap                         (picture-filename)

 busy      mask
        .
Cursor    Bitmap                         (picture-filename)
```

and so on for the other cursor names.

Desktop appearance

The following values give the dimensions of the desktop window if X.desktop is not running as a window manager.

```
desktop    x
        .
Desktop    X                             (number)

desktop    y
        .
Desktop    Y                             (number)

desktop     width
        .
Desktop     Width                        (number)

desktop    height
        .
Desktop    Height                        (number)
```

These numbers are supplied as preferences to the window manager but may be ignored.

The background patterns used by the desktop are specified by:

```
desktop       backgroundPixmap
          •
Desktop       BackgroundPixmap                     (picture-filename)

directory        backgroundPixmap
            •
Directory        BackgroundPixmap                  (picture-filename)

directory        scrollbar        backgroundPixmap
            •                 •
Directory        Scrollbar        BackgroundPixmap
                                                   (picture-filename)
```

Each item should be the name of a picture file. This picture will be tiled across the area in question. The default pictures are built into X.desktop.

Menus

The following items specify the menu drop-shadow width, the height of an ordinary dividing bar, and the height of the bar beneath the menu title.

```
              itemBar      height
          •            •
 Menu          Bar         Height                      (number:2)

              titleBar     height
          •            •
 Menu          Bar         Height                      (number:2)

              shadowWidth
          •
 Menu         ShadowWidth                              (number:2)
```

Message windows

The following items affect the visible appearance of message windows:

```
              borderWidth
          •
Message       BorderWidth                              (number:4)

              innerMargin
          •
Message       InnerMargin                              (number:10)
```

The default values are four pixels for the border of the window, and 10 pixels of space between the border and any text or the go-away button.

Launching programs

The following items control whether the cursor should be changed to the 'launch' cursor when a program is run.

```
process     launch      show
          •          •
Process     Launch      Show                          (true/false)

process     launch     cursor      data
          •          •         •
Process     Launch     Cursor     Bitmap   (cursor picture file)

process     launch     cursor      mask
          •          •         •
Process     Launch     Cursor     Bitmap   (cursor picture file)

process     launch      time
          •          •
Process     Launch      Time                           (number:3)
```

If the show option is on, then the cursor will be changed to the 'launch' cursor for the time given (in seconds). The default is for launching to be shown for three seconds, using a built-in cursor.

```
process     showBorder
          •
Process     ShowBorder                                (true/false)
```

This option indicates whether process window borders should be visible inside the process window frame.

Buttons and icons

The following items give the pictures for the buttons and icons needed to run X.desktop:

```
directory     byIcon     pixmap
            •          •
Directory     Button     Pixmap            (picture-filename)
```

for each of the buttons `byIcon`, `byName`, `close`, and `grow`.

```
 alert      goAway     pixmap
          •          •
Message     Button     Pixmap              (picture-filename)
```

for each of the messages `alert`, `fatal`, `greeting`, and `information`.

```
process      grow      pixmap
Process   •  Button  •  Pixmap                (picture-filename)
```

for each of the buttons `grow` and `close`.

```
default     pixmap
 Icon    •  Pixmap                            (picture-filename)
```

for each of the icons `default` and `newFile`.

```
process     default     pixmap
Process   •  Icon    •  Pixmap                (picture-filename)
```

The default pictures are built into X.desktop.

Message files

Message files and language support

All X.desktop messages are kept in a file which can be edited by the user making it very easy to use X.desktop in a foreign language or tailor the X.desktop messages to specific requirements.

By using the X/Open standard 'Native Language Support', X.desktop adheres to a common method for provision of language information.

Using message files

X.desktop stores all the messages it generates and uses in a special file, called the message file.

X.desktop's message files are based on the Native Language Support facility defined by the X/Open Portability Guide. If your computer supports the NLS system, then X.desktop will use it. Otherwise, X.desktop provides a similar mechanism itself. The differences between the two systems are described here.

NLS systems

On NLS systems, the message file described in this section of *X.desktop Configuration* must be converted into a special format known as an NLS catalogue. This is done with the `gencat` utility. The search mechanism described below is then used by X.desktop to find the catalogue, rather than the message file.

The format of message files accepted by `gencat` may be more complex than that described below. If so, you may make use of any facilities supported on your system as well as those described here.

Your System Administrator may have decided on a specific location to place all NLS catalogues.

Other systems

On systems that do not support NLS, X.desktop uses the message file directly, without converting it to a different form. The search mechanism described below is used to find the message file.

Message files should only use the facilities described in this section of *X.desktop Configuration* .

You may store message files anywhere on your computer, but we would suggest that those intended for general use are stored in the directory `/usr/x/lib /xdtmessages`.

Locating the message file

X.desktop looks for message files or catalogues in various places, depending on the values of two environment variables: LANG and NLSPATH.

The LANG environment variable

LANG is used to determine which of X.desktop's message files you wish to use. The X/Open rules state that LANG should be in one of three forms:

`language`

`language_territory`

`language_territory.codeset`

where *`language`* gives the name of the language that you want your messages to be in, *`territory`* is used to indicate territorial differences (for example, between UK and US 'English', or between French, Belgian, and Swiss usages in the French language), and *`codeset`* is used to select a particular character set. If any part is omitted, then a default should be used – this default may vary from system to system. All names should be in English.

For example, a particular user might set `LANG` to `french` to indicate that they want messages in French, to `french_swiss` to further indicate that they wish Swiss conventions to be used (the default on their system might be Belgian), or to `french_swiss.8859` to indicate that the message file written in the ISO character set IS8859/1 should be used.

The NLSPATH environment variable

NLSPATH should be set to a sequence of filenames, separated by colons. X.desktop will use the first of these files that it finds. The percent character is used to indicate that something should be substituted:

`%L`	the value of `LANG`
`%l`	the language element of `LANG`
`%t`	the territory element of `LANG`, if specified
`%c`	the codeset element of `LANG`, if specified
`%N`	the string `xdt`
`%%`	the percent character.

For example, suppose that LANG is set to `french_swiss` and NLSPATH is set to the list `%L:%N.cat/%l:/nls/%l/terr_%t/code_%c/prog_%N`. Then the files looked for will be:

```
french_swiss
```

```
xdt.cat/french
```

```
/nls/french/terr_swiss/code_/prog_xdt
```

The default values used by X.desktop are:

```
LANG       english
NLSPATH    /usr/x/lib/xdtmessages/%L
```

The format of the message file

The messages in the message file are divided into numbered sets, and each message is given a number within its set. This system allows related messages to be grouped together.

Each set of messages starts with a line consisting of the word `$set` followed by a space and the set number. Anything following the number is ignored. Each message then appears on a separate line, consisting of the message number followed by a space and the text of the message. The message sets and the messages within each set can be in any order, but each set must be all together – it is not possible to split up a set and have two `$set` lines with the same number.

Comments can be added to message files. A comment line starts with a dollar sign and then a space or a tab. Anything following the space or tab is then ignored. In addition, blank lines are ignored.

Here is a simple example of a message file which contains two sets with a total of five messages.

```
$ This is an example message file.
$ These two lines are ignored.

$set 5 This is message set 5.
1 This is message 1 of set 5.
8 This is message 8 of set 5.
999 This is message 999 of set 5.
$set 26
86 This is message 86 of set 26.
4  This is message 4 of set 26.
```

There are a few special characters that can be added to messages. A backslash can be used to add non-printable characters:

`\n`	new line
`\t`	tab
`\b`	backspace
`\\`	backslash

A message can be continued on more than one line by ending the first line with a backslash.

Finally, certain messages will have values, such as filenames, substituted in them. This is done by placing the string `%`*n*`$s` in the message, where *n* is a digit.

For each substitutable object, the appropriate string should occur exactly once, but if there are more than one, they may occur in any order.

The messages that have strings substituted in them are given in the following table.

Set	No	Meaning of **%1$s**	Meaning of **%2$s**	Meaning of **%3$s**
11	2	Internal information		
11	16	Internal information		
11	28	Filename	Filename	
11	36	Type of error	X request name	Failed resource (in hex)
12	11	New name entered	File being duplicated	
12	12	New name entered		
12	13	New name entered		
12	14	Name of directory		
12	15	File to be copied	Directory copied into	
12	16	File to be moved	Directory moved into	
12	17	File to be linked	Directory linked into	
12	18	File being renamed		
12	20	File being duplicated		
12	21	File being copied	File copied to	
12	23	File being renamed		
12	24	File being written to		
12	31	Filename		
12	34	Filename		
12	36	Filename		
12	37	Filename		

Notes on individual messages

Set 1	message 11	used when renaming files or entering new filenames.
	message 12	used as the name of any process which has not specified a name.
	message 13	used as the icon name of any process which has not specified an icon name.
	message 15	a string which is longer, on the screen, than the longest reasonable filename. It is never actually displayed: only its length is used.
	message 16	the name that X.desktop will be given if run under another window manager.
	message 17	the icon name that X.desktop will be given if run under another window manager.
Set 2	messages 1 and 2	the names of colours that are looked up in the server colour data base. Message 1 is used for black and message 2 for white.
Set 3	message 100	should contain a number (say 'p').
	messages 100 to 100+p	the strings that will be taken by the defaults mechanism (see *Defaults files*) to mean 'true'.
	message 200	should also contain a number (say 'q').
	messages 200 to 200+q	the strings that will be taken to mean 'false'.
Set 11	these messages are used when a fatal error occurs.	
	message 1	used for all fatal errors that prevent X.desktop from using graphics.
	message 2	used for all fatal errors in `xdtforker`.
	messages 3 and 9	used for all fatal errors that can be reported in a stop box.
Set 12	used for warnings and other messages.	
Set 12	message 1	this message is placed in front of all messages displayed in a warning box.

Set 12 messages 32 and 33 by the terms of your licence, you must preserve IXI's copyright notices in these messages. You may change any other part of them. Message 32 is used when X.desktop is running as a window manager, and message 33 otherwise.

Set 21 these messages form the contents of the desktop menu.

Set 22 these messages form the contents of the directory menu.

Command line options

The following command line options may be specified when X.desktop is run. Where they conflict with settings and values supplied by the X defaults mechanism, the command line options take precedence.

`-d` invokes testing options: for IXI use only.

`-s` invokes testing options: for IXI use only.

`-w` runs X.desktop in a window, rather than as a window manager.

`-m` runs X.desktop as a window manager, rather than in a window.

`-c` *`file`* the specified file (normally a named pipe) will be read for XCL commands at regular intervals. Each command should be terminated with a new line.

`-f` *`font`* specifies a font to be used.

`-g` *`geometry`* (implies `-w`) specifies the size of the window to be used.

`=`*`geometry`* equivalent to `-g`.

The `-m` option may not be used with `-w`, `-g`, or `=`.

The geometry option should take one of the following forms:

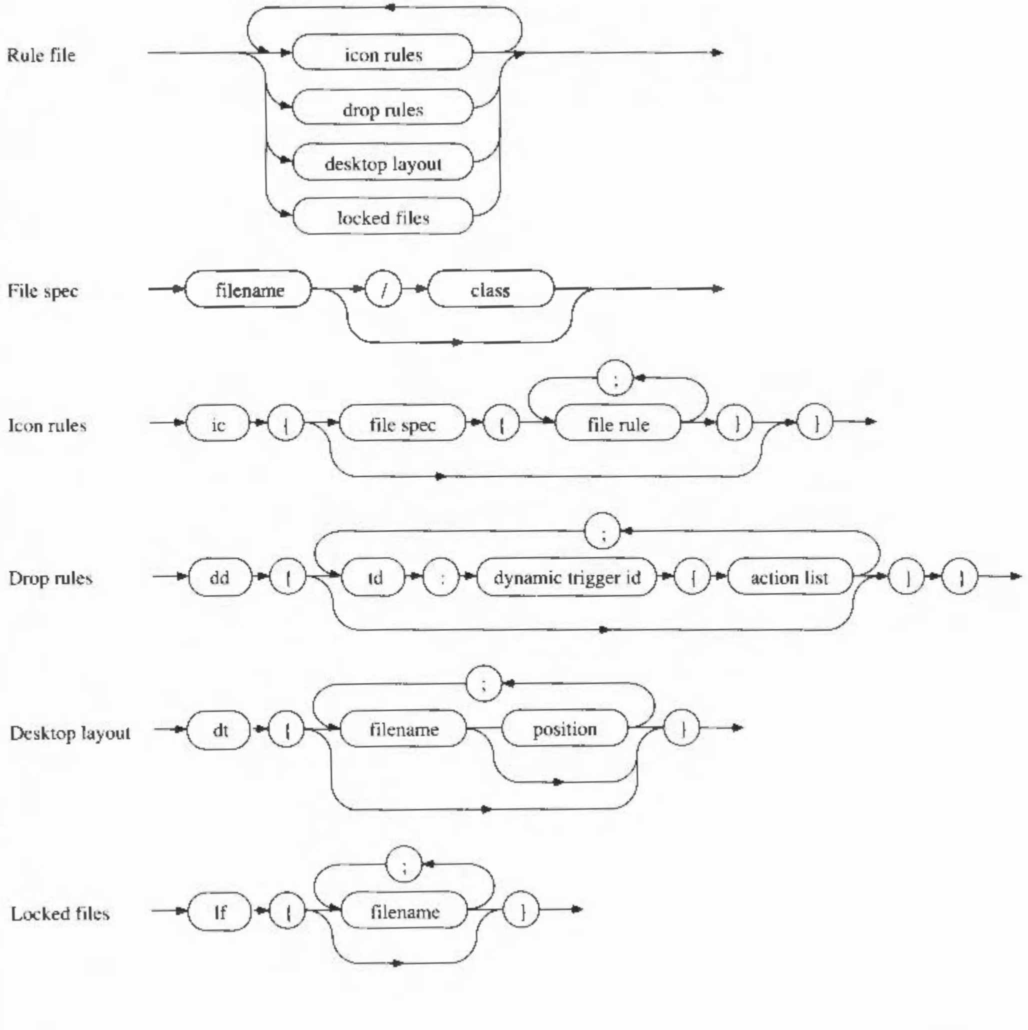
`widthxheight+x+y` For example, `500x500+100+100`

`widthxheight`

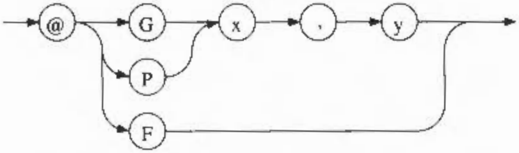
`+x+y`

Any dimensions missing will be taken from the X defaults mechanism. Either plus sign may be replaced by a minus – the number will then be the distance between the right or bottom edge of the X.desktop window and the corresponding edge of the screen.

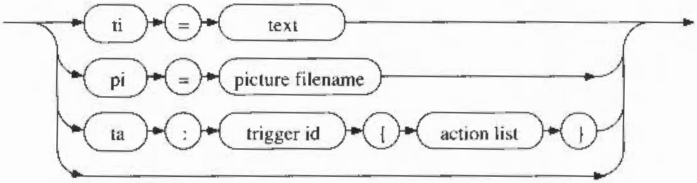
Appendix – Rule syntax diagrams



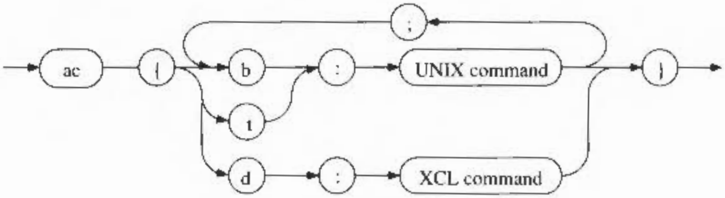
Icon position



File rule



Action list



Index

Acorn

The choice of experience.