# The Mjølner BETA System
# X Window System Libraries
Reference Manual

Mjølner Informatics Report
MIA 91-16(1.2 *update*)
August 1996

# 1 Introduction

This paper constitute an update to [MIA 91-16] version 1.2, August 1994, that describes the Mjølner BETA System X Window Libraries version 1.8 based on the basic libraries version 1.4.

The update relates to version 1.9 of the Mjølner BETA System X Window Libraries, which are based on version 1.5 of the basic libraries.

Thus an evident change is that all occurrences in the original paper of `Xt/v1.8` should read `Xt/v1.9`, and all occurrences of `basiclib/v1.4` should read `basiclib/v1.5`.

The rest of this paper concentrates on describing significant changes, and a few obvious changes like the above version numbers are left out for shortness.

# 2 Clarifications

A few typos and insufficient explanations in version 1.2 of the manual are clarified below.

## 2.1 List of Programs

The List of Programs on page iv is very inaccurate: First of all, the entries from page 58-60 regarding layout semantics should not be in the list.

Secondly all the AwEnv programs are missing from the list. The missing entries are:

## 2.2 Initialization of widgets

Section 2.2.1 in the original v1.2 of this manual briefly explains what happens when a Core widget, or specialization thereof, is initialized. The explanation is not very precise, and has caused some confusion. It has been elaborated on in the BETA FAQ, and here we will sum up.

### 2.2.1 The default Father of a Widget

Consider the following program:

```
ORIGIN '~beta/Xt/current/awenv';
--- program: descriptor ---
AwEnv
(# faculty: label
     (# init:: (# do 2-> borderwidth #) #);
   University: @box
     (# Physics, Mathematics: @faculty;
        init:: (# do Physics.init; Mathematics.init #);
     #)
do University.init;
#)
```

The idea was that a window with two labels named `Physics` and `Mathematics` should appear. But executing it will give the error message

```
Xt Error: There must be only one non-shell widget which is son
of Toplevel. The widget causing the conflict is named faculty.
```

This is because the program uses the `init` pattern of the widgets without specifying the father and name of the widgets. In the original version 1.2 of [MIA 91-16], it is briefly explained that the father widget will default to "the enclosing widget according to BETA's scope rules" (section 2.2.1).

To be precise, this is what happens: When the `init` pattern of a widget is invoked, it first checked to see if the father is `NONE`. This will be the case if no father is specified in the enter part of `init`.

If so, a search is started in the static environment of the widget pattern. If a specialization of a `Core` widget is found, this widget is used as the father. This search is continued until a pattern with no enclosing pattern is found. In this case the widget named `TopLevel` (in `xtenv`) is used as the father. The widget `TopLevel` is an instance of the pattern `TopLevelShell`, which among its characteristics has the constraint that it wants to have exactly one non-shell child.

Now consider the example program: The first thing that happens is that the `init` attribute of `University` is invoked. Since no father is specified, a search for one is started from the `University` pattern. This search finds the pattern `AwEnv(# ... #)`, which is not a `Core`, and which has no enclosing pattern. Thus `University` will get the father widget `TopLevel`.

The final binding of `University.init` then invokes `Physics.init`. `Physics` is an instance of the pattern `faculty`, which is declared in the same scope as `University`. Thus the search for a father for `Physics` is identical to the search for the father of `University`, and `Physics` also gets `TopLevel` as its father. This is when the error occurs. The reason why the name reported in the error message is `faculty` is explained in section 2.2.2 below.

Notice that it did not matter that the instantiation of the `Physics` object is done within `University`: the default father is searched for starting from the pattern declaration of the object.

In general there are three possible solutions to the problem in the example:

1. Supply the father and name when initializing the faculty widgets:

   ```
   do ("Physics", University)->Physics.init;
      ("Mathematics", University)->Mathematics.init;
   ```

   In this case, no search for a default father is needed for the faculty widgets.

2. Make (possibly empty) specializations of faculty inside University:

   ```
   Physics: @faculty(##);
   Mathematics: @faculty(##);
   ```

   Now the search for a default father of Physics will start at the pattern `faculty(##)` inside `University`, so the `University` pattern will be the first found in this search, and hence the `University` widget will become the father of the `Physics` widget. Likewise for `Mathematics`.

3. Move the declaration of the faculty pattern inside the `University` pattern. This will give the same search path as in solution 2. (Conceptually, this might also be the best place to declare `faculty` in the first place.)

The above example was a simple one. In more complicated cases, the reason for an error of this kind can be trickier to spot. If your program uses the fragment system to move declarations of useful widgets into a library, this kind of error is likely to occur.

Remember that if an instance of an unspecialized widget is used, the widget pattern being declared in, say, the `XtEnvLib` attributes slot of `xtenv`, then the search for a default father is started at the `XtEnv` pattern, and therefore no father widget is found. In this case the widget will get `TopLevel` as father. Solutions 1 or 2 above will be appropriate in these cases.

### 2.2.1 The Default Name of Widgets

The following BETA program creates a window containing "`Label`"

```
ORIGIN '~beta/Xt/current/awenv'
--- program: descriptor ---
AwEnv
(# Hello: @Label;
do Hello.init;
#)
```

whereas the following program creates a window containing "`Hello`"

```
ORIGIN '~beta/Xt/current/awenv'
--- program: descriptor ---
AwEnv
(# Hello: @Label(##);
do Hello.init;
#)
```

Here is the reason why.

The connection between the names used for widgets in BETA and the external names used in the external widgets interfaced to from BETA is that the pattern name of the BETA widget is used for the external widget name by default.

In the first example, the `Hello` widget is an instance of the pattern `Label`, and in the second example the widget is the only possible instance of the singular pattern `Label(##)`, which is named `Hello`.

The appearance of the windows in this case comes from the fact that the Athena `Label` widget uses the external name of the widget as default label-string, if it is not specified otherwise.

A variant of this problem is the case where you specify a list of widgets using the same pattern:

```
hello1, hello2: @Label(##);
```

In this case the default name will always be the first name in the list, `hello1`. To avoid this behavior, use the scheme

```
hello1: @Label(##);
hello2: @Label(##);
```

or specify the name explicitly instead.

This problem (and the solution) was mentioned without very much explanation in footnote 2 on page 54 in the original version 1.2 of [MIA 91-16]. Hopefully the above explanation clarifies the problem somewhat.

# 3 Changes in Xt version 1.9

As of version 1.9 the following changes have been introduced at interface level:

1. The pattern `FileToBitmap` in `Core` has been renamed to `BitmapFileToBitmap`. Likewise, the pattern `FileToPixmap` in `Core` has been renamed to `BitmapFileToPixmap`.

   This is for the sake of clarification only.

2. In the `athena/Prompts` hierarchy a `validate` virtual has been added.

   In `demo/awenv/getinteger` it is shown how to use this virtual to implement a dialog that only accepts integers as input.

3. *Read-only resources* have been introduced, e.g. `RoIntegerResource`. The writeable resources are specializations of the read-only resources.

4. Introduced two new resource patterns: `CharResource` and `ShortResource`.

   Except for a few left-outs, all resource-patterns that model char and short X resources, have been changed to use these new patterns as prefixes instead of the `IntegerResource`. There was a `BooleanResource` in `Xt/v1.8`, but it was simply implemented as an `IntegerResource`. This has been fixed as part of the resource-cleanup.

5. Two directories have been added: `misc` and `demo/misc`. At the moment these only contain a small interface to `xterm` windows: You can set the title and icon name of the `xterm` you are executing in, and you can start another UNIX process in a separate `xterm` window. The demo in `demo/misc/xtermdemo` shows examples of both.

# References

| | |
|---|---|
| [MIA 91-16] | Mjølner Informatics: *The Mjølner BETA System—X Window System Libraries, Reference Manual,* MjølnerInformatics Report MIA 91-16. |