USER'S GUIDE

CAPBAK/X

Version 5.2

Capture / Playback System for X Windows

SOFTWARE RESEARCH, INC.

This document property of:

Name:________________________________

Company:______________________________

Address:______________________________

____

Phone__________________________________

SOFTWARE RESEARCH, INC.

1663 Mission Street, Suite 400

San Francisco, CA 94103

Tel: (415) 861-2800

Toll Free: (800) 942-SOFT

Fax: (415) 861-9801

E-mail: support@soft.com

http://www.soft.com

Table of Contents

List of Figures

Preface

Congratulations!

By choosing the TestWorks integrated suite of testing tools, you have taken the first step in bringing your application to the highest possible level of quality.

Software testing and quality assurance, while becoming more important in today's competitive marketplace, can dominate your resources and delay your product release. By automating the testing process, you can assure the quality of your product without needlessly depleting your resources.

Software Research, Inc. believes strongly in automated software testing. It is our goal to bring your product as close to flawlessness as possible. Our leading-edge testing techniques and coverage assurance methods are designed to give you the greatest insight into your source code.

TestWorks is the most complete solution available, with full-featured regression testing, coverage analyzers, and metric tools.

Audience

This manual is intended for software testers who are using *CAPBAK/X* tools. You should be familiar with the X Window System and your workstation.

Contents of Chapters

Chapter 1	*INTRODUCTION* provides basic capture / playback terminology and specific information on TrueTime and ObjectMode recording.
Chapter 2	*QUICK START* is a tutorial for basic recordings in TrueTime, ObjectMode and Mixed Mode using *CAPBAK/X*
Chapter 3	*CAPBAK/X GRAPHICAL USER INTERFACE* provides a summary of *CAPBAK/X* windows, menus, and commands.
Chapter 4	*INVOKING CAPBAK/X* shows you how to invoke *CAPBAK/X*.
Chapter 5	*SETUP FOR OBJECTMODE RECORDING* provides information on the setup needed for ObjectMode (widget) recording with *CAPBAK/X*.
Chapter 6	*PLAYBACK SYNCHRONIZATION ISSUES* provides information on cause and resolution of playback synchronization issues.
Chapter 7	*OPTICAL CHARACTER RECOGNITION (OCR) CAPABILITY* provides information on OCR which extends the life of keysave files by tolerating minor application changes, extracting characters and finding the location of a character string.
Chapter 8	*XVIRTUAL DISPLAY SYSTEM* provides information on the operation of Xvirtual Display.

Typefaces

The typographical conventions used in this manual:

boldface	Introduces or emphasizes a term that refers to TestWorks' window, its sub-menus and its options.
italics	Indicates the names of files, directories, pathnames, variables, and attributes. Italics is also used for manual and book titles.
"Double Quotation Marks"	Indicates chapter titles and sections. Words with special meanings may also be set apart with double quotation marks the first time they are used.
`courier`	Indicates system output such as error messages, system hints, file output, and *CAPBAK/X*'s keysave file language.
`Boldface Courier`	Indicates any command or data input that you are directed to type. For example, prompts and invocation commands are in this text. (For instance, **stw** invokes TestWorks.)

CHAPTER 1

Introduction

INFORMATION: Basic capture / playback terminology, specific information on True-Time and ObjectMode recording. The rationale behind both methods is explained.
SEE ALSO: Chapter 2, “Quick Start”; Chapter 6, “Playback Synchronization Issues.”
LEVEL: All users. Those familiar with these concepts may skip to Chapter 2.

1.1 Introduction to Capture/Playback

Testing the graphical user interface (GUI) part of a product can be very complex, and when the GUI is part of a client-server application in which more than one machine is involved, it can be even more complicated. Some test-tool suppliers would have you believe that you can *never* do tests on GUIs without ObjectMode testing, and other organizations would have you believe that only “TrueTime" (100% realistic) recording and playback are needed. In practice, *both* are required, both are useful, and both methods have advantages and limitations.

1.2 TrueTime Tests

Since the mid-1980's, when SR introduced automated capture and replay technology, first for MS-DOS and then for UNIX serial-port environments, the main technology for test capture / playback has been “True-Time” recording of test sessions. Keyboard and mouse activity events are recorded and played back in a way that maintains time- and position-based faithfulness to exactly what the user entered.

The preservation of TrueTime provides for a reliable playback because, it is fairly assumed, the user will not make a recording of something that is unrealistic. Hence, during playback, so long as the machine has enough capability, playback would not “fail to synchronize."

A failure to synchronize during playback causes the test recording to abort, and typically signals that the application has changed in a way that the test has detected. The problem is that in some cases, when the application hasn't changed, the failed test would imply that it had - incorrectly. Too many such false-negative results would tend to indicate that the test suite is unreliable.

1.2.1 Synchronization Aspects

Done correctly, and with the correct kinds of playback synchronization enhancements in place, TrueTime-based testing can be extremely effective. Generally, such tests are very precise; this means that a TrueTime test will FAIL - i.e. correctly identify a defect - even if the smallest product change occurs.

In X Windows there are some special considerations for TrueTime testing, such as how to relocate windows to the original positions, etc. These matters are discussed in more detail elsewhere (see also “Synchronization Issues”), but it suffices to recognize that most operational objections are easily overcome.

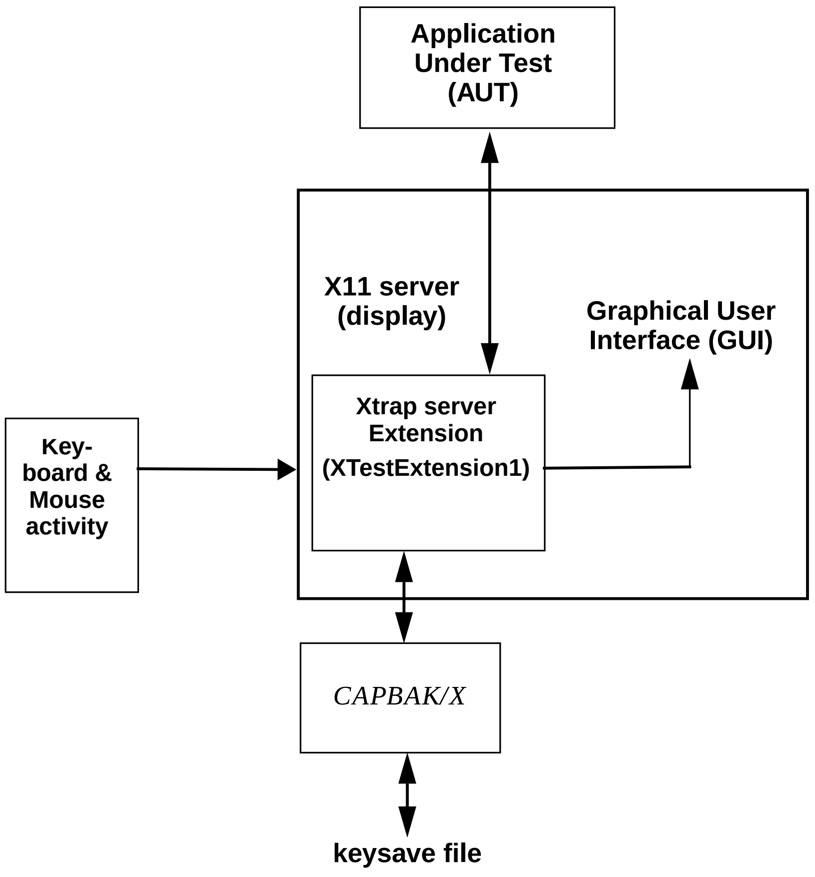

Application Under Test (AUT)
X11 server (display)
Graphical User Interface (GUI)
Xtrap server Extension (XTestExtension1)
Key-board & Mouse activity
CAPBAK/X
keysave file

FIGURE 1 TrueTime Recording Scenario

1.3 ObjectMode (“OO”) Tests

More recently, with the increased need for tests that execute without fail on multiple platforms (see below), SR's response has been to combine TrueTime capability with a different kind of test recording: ObjectMode (also known as “OO” or “Object-Oriented” testing). This method is based on use of special techniques to record which visual image, or object, is used, along with when and where it is used. TrueTime and ObjectMode can be used independently, or combined to deliver the best features of each.

ObjectMode tests play back more reliably than TrueTime tests on multiple platforms because such tests are inherently less sensitive to tiny changes in the application window features, border width, placement, color, etc.

In ObjectMode testing, the application’s widgets are directly activated during test playbacks, so that their precise X and Y coordinates are not crucial to the success of the test. Because of this, ObjectMode’s advantages are its conceptual simplicity and cross-platform portability.

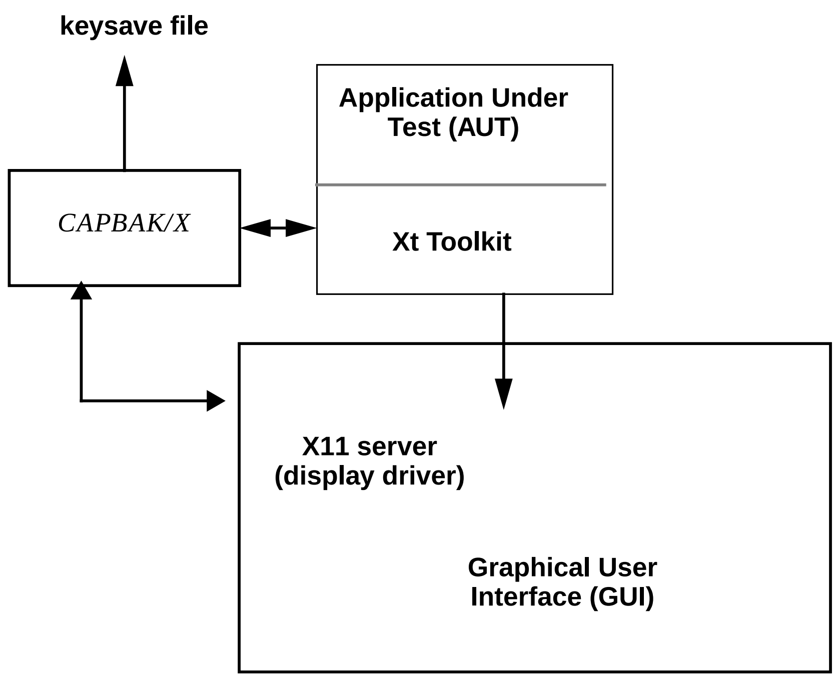

keysave file
Application Under Test (AUT)
CAPBAK/X
Xt Toolkit
X11 server (display driver)
Graphical User Interface (GUI)

FIGURE 2 ObjectMode Recording Scenario

1.3.1 Where the Problems Lie

Almost all ObjectMode tests need a semi-invasive technique that may introduce its own kind of problems to the testing process. To run in ObjectMode, a capture / playback system *must* instrument the underlying windowing library (typically it is the Xt library for Motif applications).

When you run your application tests this way there is the risk that, because the application is not really the same one that you are going to be shipping, you can't really be sure that you've tested the same software the customer gets. In other words, there is a risk that such invasivity - done to simplify the testing process - will: (a) cause errors that are present in the application to be missed by the tests (false-negative tests); and / or, (b) find errors that aren't really present (false-positive tests). Neither of these test outcomes build confidence in the application's quality if they occur often.

Typically, one must link the application to the special Xt library. On some platforms this is done at runtime, but on some platforms this “dynamic linking” is not available and a separate build is required. Separate builds introduce another source of potential error.

Another concern is that some approaches, in which the user “programs” the way the application is going to be exercised, make the testing process as vulnerable to error as the programming process itself. This is because tests are programmed to exercise what the tester believes is *supposed* to have been implemented. If the test misses a required feature, it will succeed incorrectly, because that feature will never have been tried. Even worse, if the tester’s program incorrectly exercises a feature - for example, if it pushes a button that is not actually visible on the screen - then the test will also succeed but it will have done so using “illegal” means.

1.4 Detailed Analysis

This section analyzes these points (and a few more) in more detail, addressing the advantages and disadvantages of each method.

1.4.1 TrueTime Advantages

Reproducibility TrueTime (TT) tests have excellent reproducibility: they confirm identical operation before / after a software product change, and typically detect the smallest change. The user can be assured that there is an *exact* match between what was recorded and what is played back.

Realism TT tests support realistic load generation. For example, if you are using the background mode X11 server, **Xvirtual**, you know the load you are imposing is just like a real user's load. It has the same keystroke rates and inter-key delays, and the exact same excursion activity, and clicks / drags properties.

Vectored Playback

Playback of input to something graphical, e.g. **idraw**, will be completely accurate. If the application accepts mouse vectors then TrueTime is *needed* for specific positioning and speed of mouse movement as both can affect what is drawn.

Missing Feature Detection

Good protection if you delete a feature: the test will almost certainly fail because the recorded test asks for something that is no longer there to be played back.

Added Feature/Modified GUI Detection

Good protection if you add a feature, which generally implies a change to the layout of the GUI. This will be detected as the test goes to the previous location of a UI object and tries to invoke it.

Screen-Position Dependent

No reprogramming if you change the name of a button: things are position-dependent. The action will continue to work as it did, but if required the name change can still be detected via screen comparisons.

Highly Sensitive Detector

TT testing is highly sensitive to the smallest changes! Every change is flagged as a test failure. This gives the tester the ability to determine the significance of the change rather than having the tool do it. However, this can also be a disadvantage.

1.4.2 TrueTime Disadvantages

Brittle Tests Tests fail for trivial reasons and, short of re-recording the test, or by use of OCR, there's little you can do about it. This is the disadvantage of highly sensitive checking, once the tester has determined the change is not significant.

System Dependency

If your window border offsets change, the whole suite could fail, necessitating more work. This goes back to the requirement that all tests need to be run from a known state, and with TT this includes the state of the operating environment.

Cross-Platform Testing

Running the same script on multiple platforms will not necessarily work. Differences between the platforms such as screen geometry, window system look and feel, and font support can all affect the ability of a TT test to run across different platforms.

Depends on Reproducible Behavior of System Under Test

TT testing may not work, if the system under test does not have the property that:

same initial state + the same inputs = same final state

Flight Simulator is a good example; a recorded flight generally won't reproduce with *CAPBAK/X* because the internal state is updated much faster than the resolution of the playback process (1 msec). Eventually the internal state differs enough so that the roundoff error in input event timing causes a slight state change. The plane you've recorded your flight on almost always crashes.

1.4.3 ObjectMode Advantages

Cross-Platform Testing

A test on one application will exercise the same features regardless of the platform. ObjectMode tests are independent of screen geometry, window manager features / look, and font availability.

Less Brittle Tests The same test passes if all the buttons have the same name but are in different location. This extends the life of a test due to changes from release to release of the screen layout.

Missing Feature Detection

There is good protection if you delete a feature. The test will fail as it attempts to provide input to a button or other object that is no longer present in the application.

Changed Feature Detection

There is good protection for a changed feature as the test will provide the old inputs into the object. This should result in improper outcomes for the test.

Application Change Insensitivity

ObjectMode tests are less sensitive to changes. Hence, they are more “forgiving” of any noted regressions. This is particularly useful for an application going through a series of user interface modifications.

Reprogramming Simplified

Test scripts are easy to modify as they are not dependent on location, but simply on the object that needs to be exercised and the action necessary on that object. This allows for easy-to-read scripts that look like a series of function calls.

1.4.4 ObjectMode Disadvantages

Impossible Tests Possible

ObjectMode tests can run applications in a way that a user can not. The ObjectMode test can push an invisible button, input to an object in an iconified window, or provide input to a window obscured by other windows or off-screen. All of these actions are unrealistic and do not provide for user-level testing.

Extra Feature Detection

You have no protection if you add a button to a GUI. The test won't know something was added as the change to the user interface is not detected by the ObjectMode test.

Invasivity for Statically Linked Applications

ObjectMode testing is supported using a modified toolkit which is either dynamically linked in at run-time or statically linked at compile time. This means that the tested application is different from what will actually be shipped to the user. This change should not be significant, but does add a risk to the released application.

Reprogramming Required

Reprogramming will be needed if you change the name of an object, such as a button. These programming changes are straightforward, but must all be made for the test to run correctly.

Unrealistic Load Generation

Playback in Xvirtual will work but does not impose a 100% realistic user-like load. Recordings or programmed scripts in ObjectMode do not contain timing information, so user interactions can at best be estimated.

1.5 Conclusion

CAPBAK/X supports software testing by recording what a user / tester does in such a way that the recording can be played back to determine if (a) the application software continues to operate as before or (b) has somehow changed. There are a variety of modes in which capture / playback tools can be operated. It is important to understand the strengths and weaknesses of each of these modes.

CAPBAK/X supports TrueTime, Character Recognition (OCR) and ObjectMode recording. Because testing is not a single task, but a process that goes on throughout the life of an application, each of these modes has benefits at different times during that process.

It is clear that no single GUI test method will suffice to handle every situation, but it appears that the mixture of TrueTime and ObjectMode testing gives the software tester the best possible chance to build tests that are:

- Non-invasive if they have to be (using TrueTime recording);
- Based on what the GUI really shows (using the ability to derive tests from hands-on recording rather than programmatically);
- Capable of testing graphics - where TrueTime mode is critical - as well as testing the GUIs that drive the graphics; and,
- Reliable and flexible enough to provide increased confidence in delivered product quality.

Software testing is a complicated enough problem without burdening the tester with the need to choose between two effective test modes. Therefore, testers should use both modes, picking the best one for each particular test.

CHAPTER 2

Quick Start

INFORMATION: Tutorial for basic recordings in TrueTime, ObjectMode and Mixed Mode using *CAPBAK/X*.
SEE ALSO: Chapter 1, “Introduction”; Chapter 5, “Setup for ObjectMode Recording”; Chapter 6, “Playback Synchronization Issues”; Chapter 7, “Optical Character Recognition (OCR).”
LEVEL: This chapter is intended for all users.

2.1 TrueTime Recording

The TrueTime recording mode records and plays back test sessions with temporal and positional precision, exactly mimicking all of the user’s interactions with mouse and keyboard during the test session. TrueTime testing will respond to very slight changes in the application under test.

ObjectMode recording is described in another section (See Section 2.2 - “Making Object Mode and Mixed-Mode Recordings” on page 31.).

2.1.1 Setting Up *CAPBAK/X*

Make sure you are in the X Windows System running a window manager (e.g., **mwm** or **olwm**). Consult your system administrator if you are unsure about your windowing environment setup.

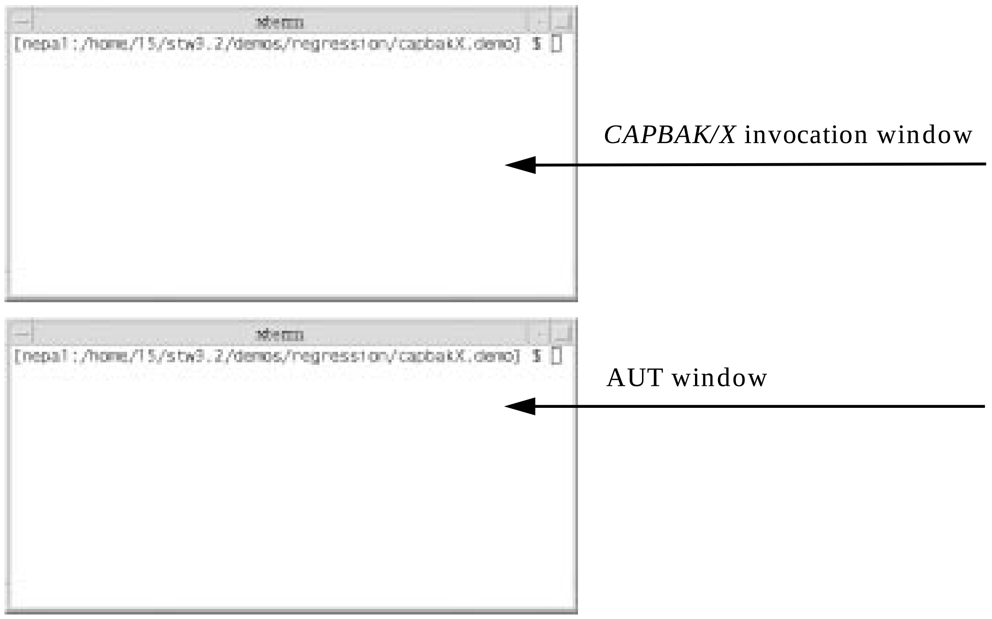
If you are running Sparc OpenWindows, be sure to invoke the X11 / NeWS server by entering:

```
openwin -defeateventsecurity
```

2.1.1.1 Organizing Your Display

In any directory where you have **write** permission, open two xterm windows from an existing window. One xterm window will be the application-under-test (AUT) window, and the other will serve as the *CAPBAK/X* invocation window.

The screen display should look like this:



```
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $
```

```
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $
```

FIGURE 3 Setting Up *CAPBAK/X*

2.1.2 Invoking *CAPBAK/X*

To invoke *CAPBAK /X*:

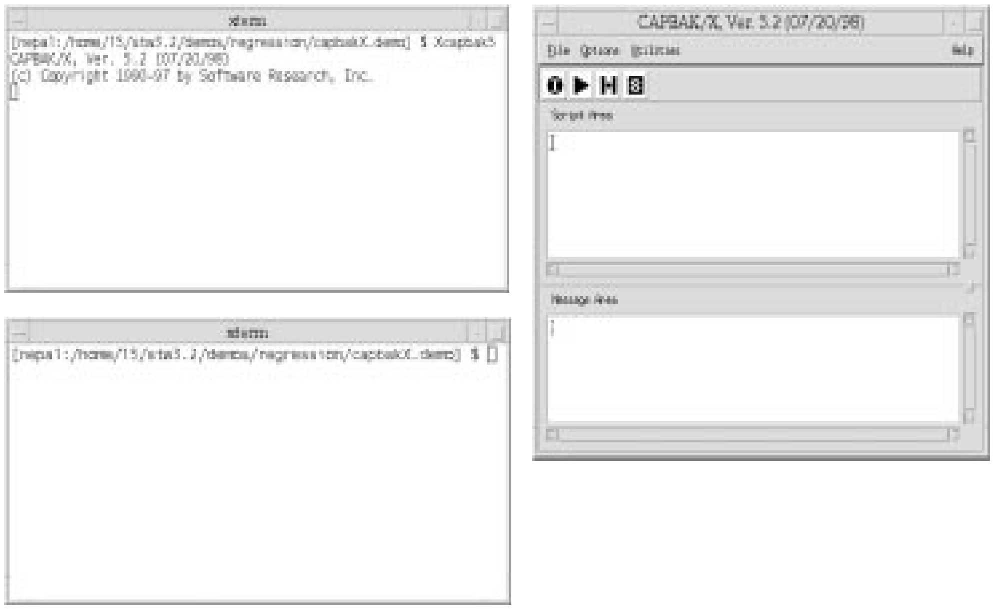
1. Activate the *CAPBAK/X* invocation window by clicking it. This window will serve as the control window for your session. Status messages and warnings will be displayed in this window.
2. Start *CAPBAK /X* from your working directory by typing:

   `Xcapbak5`

The *CAPBAK/X* main control window pops up. Move it to the upper right of the screen. Its control panel allows you to record and play back sessions, and also to initiate its other utilities, e.g. viewing captured images.

3. If you want to start over, you can terminate *CAPBAK/X* by clicking on the **File** menu and selecting the **Exit** option.

After *CAPBAK/X* is invoked and the **Main** window is relocated, the screen display should look like this:



```
[nepal:/home/15/stw5.2/demos/regression/capbakX.demo] $ Xcapbak5
CAPBAK/X, Ver. 5.2 (07/20/98)
(C) Copyright 1990-97 by Software Research, Inc.
```

```
[nepal:/home/15/stw5.2/demos/regression/capbakX.demo] $
```

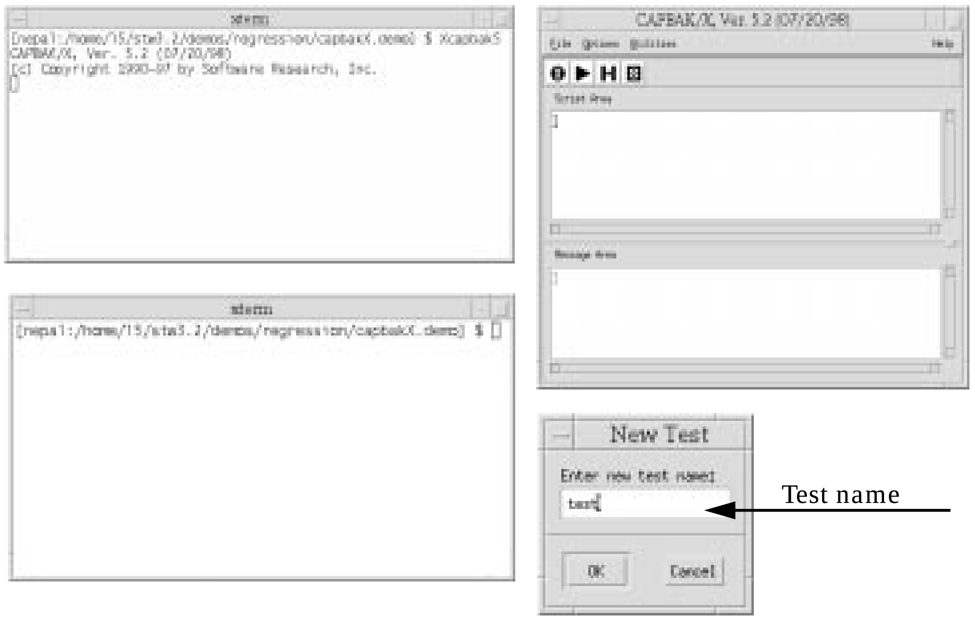
FIGURE 4 Invoking *CAPBAK/X*

2.1.3 Naming a Test

Before recording, you must name a test file where your inputs will be saved. The test file will contain an editable script of all mouse movements, keystrokes, and any screen capture input statements executed during your test session. To name a test using the Record **(I)** button:

1. Click on the Record (**I**) button.
2. A file selection window pops up over the *CAPBAK/X* main invocation window.
3. Type **`test`** in the entry box. All test names are automatically given a `.ksv` extension when you **Save** them at the end of the session. Test names are limited to valid C function names and must begin with an alpha character (no digits or special characters).
4. Click on **OK** or press the **Enter** key, after you type in the test name.

When you are assigning a test name, your display should look like this:



```
[nepa1:/home/15/sta5.2/demos/regression/capbakX.demo] $ Xcapbak5
CAPBAK/X, Ver. 5.2 (07/20/98)
(c) Copyright 1990-97 by Software Research, Inc.
```

```
[nepa1:/home/15/sta5.2/demos/regression/capbakX.demo] $
```

FIGURE 5 Naming a Keysave File

For more information on creating keysave files, refer to Chapter 8, section 8.9.2.1.

2.1.4 Meet Motifburger

2.1.4.1 Using Motifburger as a Sample Application Under Test

This section will familiarize you with the sample Application-Under-Test (AUT). Motifburger is a sample application provided as part of OSF/Motif Distribution and demonstrates some GUI objects available. The files for Motifburger are located in:

`$SR/demos/regression/capbakX.demo/`.

For more information on these files, open the `README`.Motifbur file in the same directory.

2.1.4.2 Invoking Motifburger

To invoke the sample AUT, type in the AUT xterm window:

```
motifbur
```

The Motifburger main window pops up.

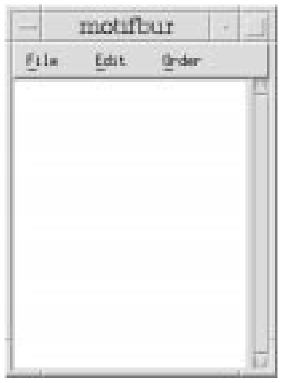


FIGURE 6 Motifburger Main Window

Note: If a message is displayed,
"`Application Under Test 'motifbur' is linked for widget-level testing.`"
you are now recordable in Object Mode. Please refer to Section 2.2.2

2.1.4.3 File Pull-Down Menu

To exit Motifburger, click the File pull-down menu and select Quit.

2.1.4.4 Order Pull-Down Menu

Click on the **Order** menu. There are three options: **Create order box...**, **Cancel Order,** and **Submit Order**. Click **Create order box...**

The **Motifburger Order Entry Box** appears, with three columns: **Hamburgers**, **Fries** and **Drinks**:

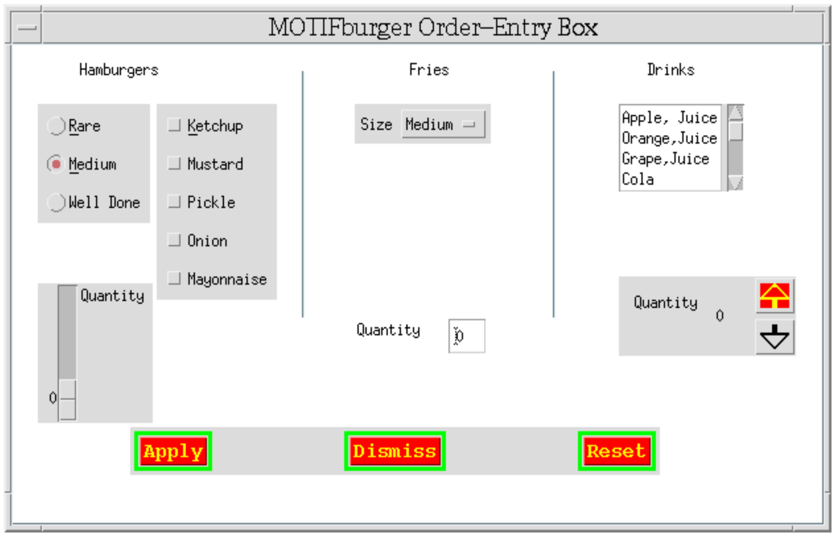


FIGURE 7 Motifburger Order Entry Box

Hamburgers

Click on how you want your hamburger cooked; on the far left side of the box under **Hamburgers**, pick **Rare**, **Medium** or **Well Done**.

Move the **Quantity** slide counter (at the bottom left) to indicate the number of hamburgers. Moving the slide counter up increases the quantity, which is displayed numerically to the left of the slide counter.

Select your condiments in the **Hamburgers** column, where there are selection buttons for **Ketchup**, **Mustard**, etc.

Fries

Move the mouse to the **Fries** column in the middle of the screen and click on the **Size** pull-down menu; the fries range from **Tiny** to **Huge**.

Use the **Quantity** box to select a quantity of fries: in this case you must click the pointer in the dialogue box and type the number.

Drinks

Next, go to the column listing **Drinks** on the right-hand side of the box. Select your drink by clicking on it.

Use the up arrow and down arrow buttons to indicate how many drinks you want.

After you have made your selections, move the mouse pointer to one of the three buttons at the bottom of the screen:

- If you click **Apply**, all the actions in the **Order Entry Box** will be saved in the main window.
- If you click **Reset**, all the selected items will be de-selected and the fields returned to default settings.
- **Dismiss** will close the dialog box and take you back to the Motif-burger main window. There you can choose whether to save your selections to a file or not.

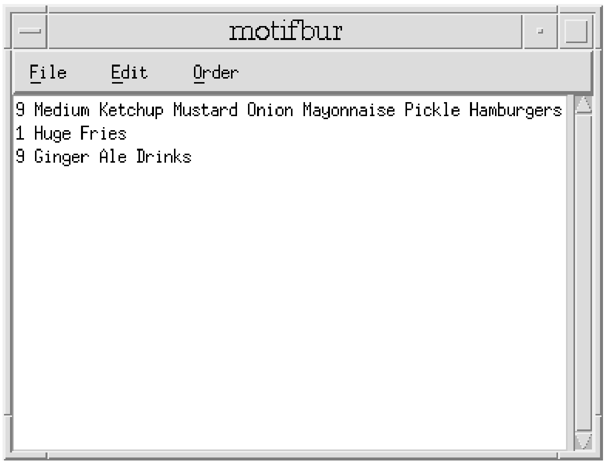
Cancel Order

Any actions from the **Motifburger Order Entry Box** will not be saved to a file.

Submit Order

The actions from the **Motifburger Order Entry Box** window will be saved to a file.

2.1.4.5 Motifburger Main Window After Ordering



```
9 Medium Ketchup Mustard Onion Mayonnaise Pickle Hamburgers
1 Huge Fries
9 Ginger Ale Drinks
```

FIGURE 8 Motifburger Main Window after Ordering

The **Motifburger** main window accounts for actions during the test; in Figure 8, the user selected nine ketchup mustard onion mayonnaise pickle hamburgers cooked medium, one huge fries and nine ginger ales in the **Motifburger Order Entry Box**, and then clicked **Apply**.

You have now seen how the sample application under test works. Instructions for actual testing are next. Before beginning the actual recording, close the **Motifburger Order Entry Box** window by clicking on **Dismiss**. Then select **Quit** under **File** pull-down menu in the **Motif-burger** window.

2.1.5 Starting a TrueTime Recording

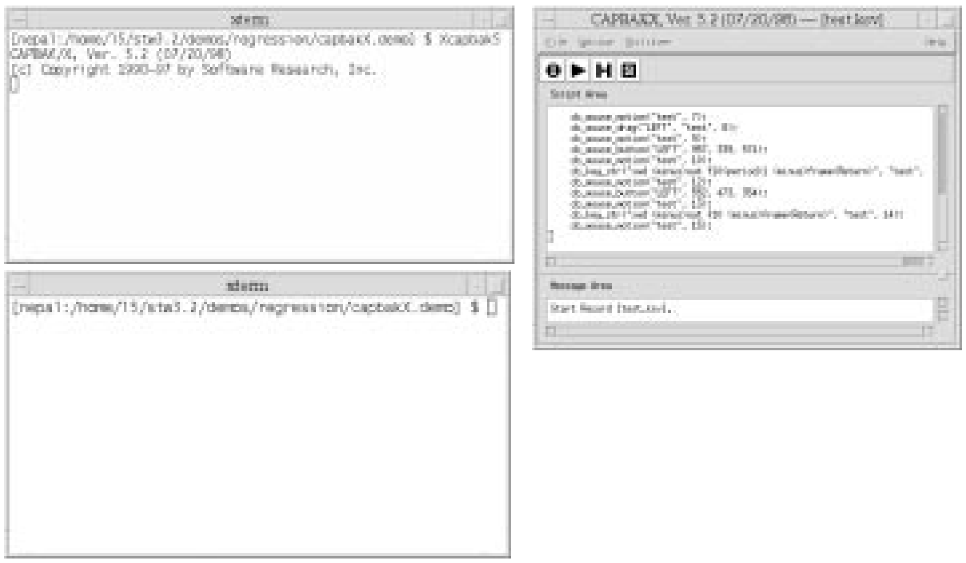
1. Click on the recording button (the control panel button labeled **I**).
2. The *CAPBAK/X* status window prompts you with the following message in the Message Area (lower half of *CAPBAK* window):

   `Start Record.`
3. You are now in recording mode. Any keyboard and mouse input you enter is recorded.

   For this session, Motifburger will be the application tested.

Note: If the Status message reads "`Could not initialize...`" refer to section 5.2 of the Installation Instructions, "Platform-Dependent Instructions."

When you are initiating a recording, the display should look like this:



```
[nepal:/home/15/sta5.2/demos/regression/capbakX.demo] $ Xcapbak5
CAPBAK/X, Ver. 5.2 (07/20/98)
[c] Copyright 1990-97 by Software Research, Inc.
```

```
[nepal:/home/15/sta5.2/demos/regression/capbakX.demo] $
```

```
[illegible]
```

```
Start Record [test.krv].
```

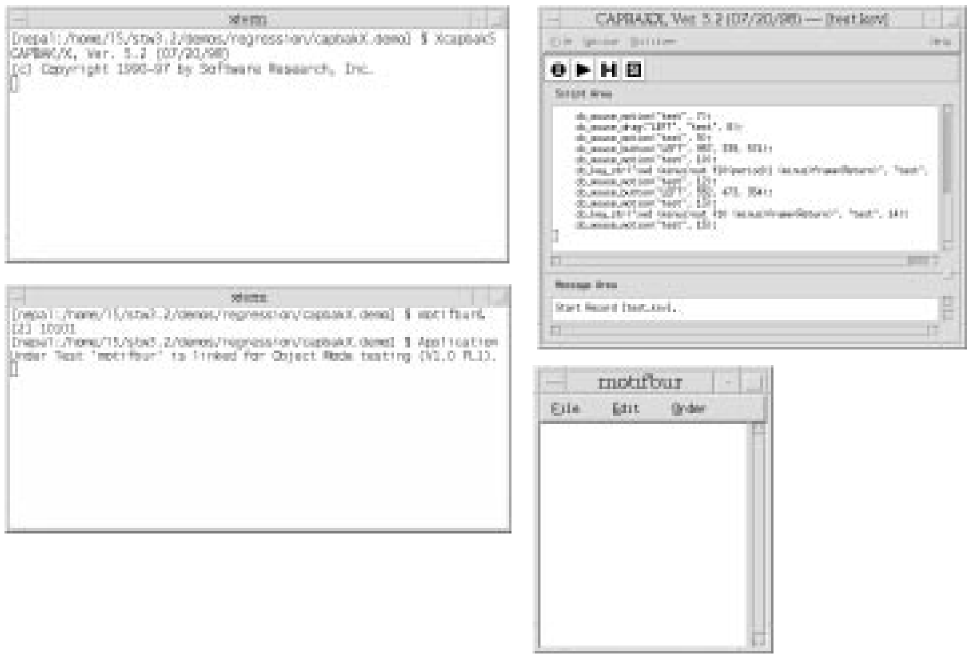
FIGURE 9 Beginning a TrueTime Recording Session

2.1.6 Creating a Test

For this test session, make a recording using Motifburger. Start the session by executing the application under test (AUT) from your AUT xterm window. Follow these steps:

1. Move the mouse around.
2. Activate the lower left window by clicking on it. This window serves as our AUT activation window.
3. Type in **`motifburger`** and then **Enter**.
4. Move the Motifburger window below your *CAPBAK/X* window.
5. Use the mouse button to open Motifburger's **Create order box.** Make some selections in the **Create order box.**

When you begin the Motifburger test session, your display should look like this:



```
[nepal:/home/15/sbw3.2/demos/regression/capbakX.demo] $ XcapbakS
CAPBAK/X, Ver. 5.2 (07/20/98)
(c) Copyright 1990-97 by Software Research, Inc.
```

```
[nepal:/home/15/sbw3.2/demos/regression/capbakX.demo] $ motifbur&
[2] 10001
[nepal:/home/15/sbw3.2/demos/regression/capbakX.demo] $ Application
Under Test 'motifbur' is linked for Object Mode testing (V1.0 FL1).
```

```
Start Record [test.kov].
```

FIGURE 10 Creating a Test

2.1.7 Capturing a Window Image

During a recording, you can command *CAPBAK/X* to place comments in the keysave file, define windows or images for synchronization during playback, or capture windows, images or the entire screen. These activities can be performed either by using options from the **Hotkey** window or by using your keyboard's function keys.

During this recording, you will capture the Motifburger window with your mouse. These instructions show you how to capture a window using either the **Hotkey** window or the function keys.

2.1.7.1 Using the Hotkey Window

1. Press the **F2** function key when you are ready to capture the Motifburger window. Recording pauses and the **Hotkey** window pops up.
2. The *CAPBAK/X* window's **Message area** prompts you with this message:

   ```
   Command Mode. Select option from menu.
   ```
3. Click on the **Save Image Window** button. The mouse pointer turns into a cross-hairs symbol.
4. Move the mouse cross-hairs to the Motifburger window and then click and release the mouse button. Avoid clicking on any of Motifburger's buttons.

   The system signals capture with a bell, and the *CAPBAK/X* window's **Message area** reads:

   ```
   Mouse window image saving...complete.
   ```

Note: Make sure that you close the **Hotkey** window prior to any further recording; otherwise, anything else you do will **not** be recorded.

2.1.7.2 Using Function Keys

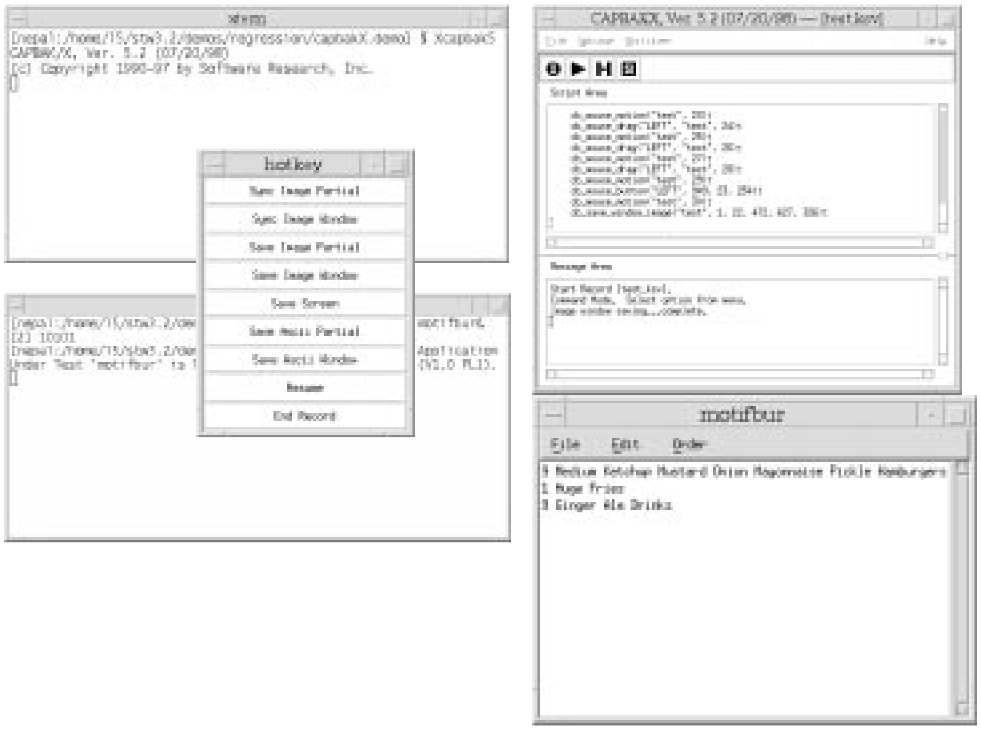
1. Move the mouse pointer over the Motifburger window.
2. Press the **F6** function key.

   Recording pauses as the image is being captured. The system signals capture completion with a bell and the status window gives you the following message:

   ```
   Mouse window image saving...complete.
   ```

   Using either method of image capture, the Motifburger window image is automatically saved as *test.001* in your `cbx_db/CBX/BSL` subdirectory. This file serves as your baseline image, which you can compare with response images captured during playback sessions.

While a window is being saved with the **Hotkey** window, the display should resemble the following:



```
[nepal:/home/15/sbw3.2/demos/regression/capbakX.demo] $ Xcapbak5
CAPBAK/X, Ver. 5.2 (07/20/98)
(c) Copyright 1990-97 by Software Research, Inc.
```

```
[nepal:/home/15/sbw3.2/dem                          motifbur&
[2] 10001
[nepal:/home/15/sbw3.2/dem                          Application
Under Test 'motifbur' is [illegible]               (V1.0 PL1).
```

```
do_mouse_motion("test", [illegible]);
do_mouse_drag("LEFT", "test", [illegible]);
do_mouse_motion("test", [illegible]);
do_mouse_drag("LEFT", "test", [illegible]);
do_mouse_motion("test", [illegible]);
do_mouse_drag("LEFT", "test", [illegible]);
do_mouse_motion("test", [illegible]);
do_mouse_button("LEFT", [illegible], 21, 254);
do_mouse_motion("test", [illegible]);
do_save_window_image("test", 1, 22, 411, 427, 336);
```

```
Start Record [test.kev].
Command Mode. Select option from menu.
Image window saving...complete.
```

```
9 Medium Ketchup Mustard Onion Mayonnaise Pickle Hamburgers
1 Huge Fries
3 Ginger Ale Drinks
```

FIGURE 11 Capturing a Window Using the Hotkey Menu

2.1.8 Completing a TrueTime Recording Session

You can end your session by using the **Hotkey** window or the **F9** function key, and then clearing the screen.

2.1.8.1 Using the Hotkey Window

1. If you captured the Motifburger window using the **Hotkey** window, this window should still be displayed on your screen. If you saved the Motifburger window using the **F6** function key, you can still use the **Hotkey** window. Simply press **F2** and it pop ups.
2. Click on the **End Record** button. The *CAPBAK/X* main invocation status window signals the end of a recording session with the message: `End Record.`

2.1.8.2 Using the Function Keys

1. When you are ready to stop recording, simply press the **F9** function key.
2. The **Record /Play**'s status window signals the end of a recording session with the message: `End Record`.

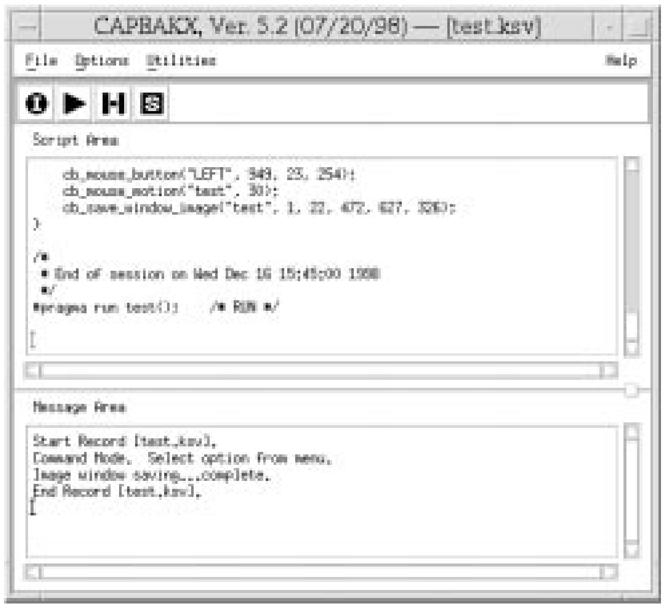
2.1.8.3 Saving the Test

To save the results of this test, pull down the **File** menu and click on **Save**. The file name will be saved in the current working directory with a `.ksv` extension. Otherwise, your work will not be saved.

2.1.8.4 Clearing the Display

1. Close Motifburger by clicking on its upper left-hand button so that its window menu appears.
2. Drag the mouse to **Close** and let go of the mouse button. The window closes.

When a recording is ending, the *CAPBAK/X* panel should look like this:



```
    cb_mouse_button("LEFT", 349, 23, 254);
    cb_mouse_motion("test", 30);
    cb_save_window_image("test", 1, 22, 472, 627, 326);
}

/*
 * End of session on Wed Dec 16 15:45:00 1998
 */
#pragma run test();    /* RUN */
```

```
Start Record [test.ksv].
Command Mode.  Select option from menu.
Image window saving...complete.
End Record [test.ksv].
```

FIGURE 12 Completing a Recording Session

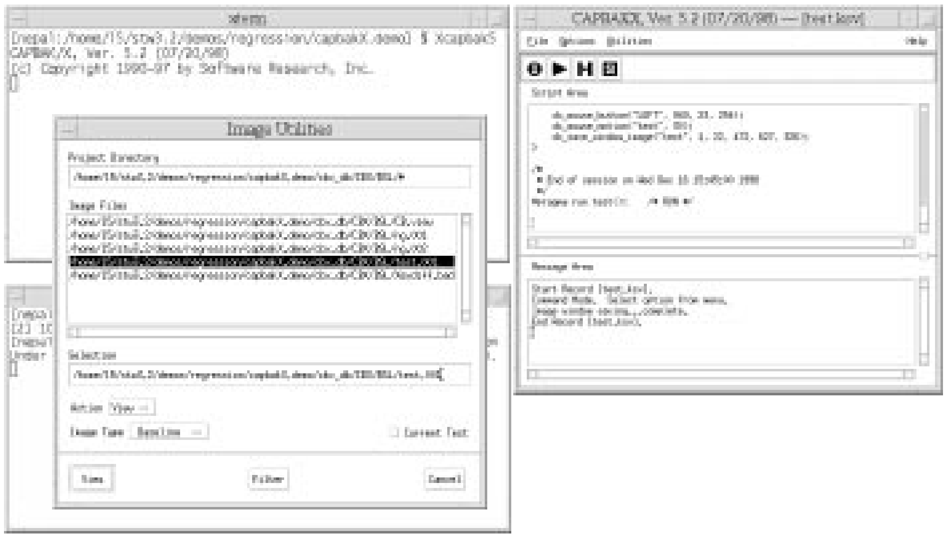
2.1.9 Initializing the View Images Window

You can verify your recording's success by looking at the Motifburger window image you saved with the **View Images** utility during the recording session. To invoke it:

1. Click the **Utilities** menu and select the **Image Utilities** submenu.

   The **View Image**s window pops up.
2. Move the **View Images** window below the *CAPBAK/X* main invocation window.

When the **View Images** window is initialized, the display should look like this:



```
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $ Xcapbak5
CAPBAK/X, Ver. 5.2 (07/20/98)
(c) Copyright 1990-97 by Software Research, Inc.
```

```
/*
 * End of session on Wed Dec 15 [illegible] 1998
 */
```

```
Start Record [test.ksv].
Command Mode. Select option from menu.
Image window saving...complete.
End Record [test.ksv].
```

FIGURE 13 Initializing the View Images Window

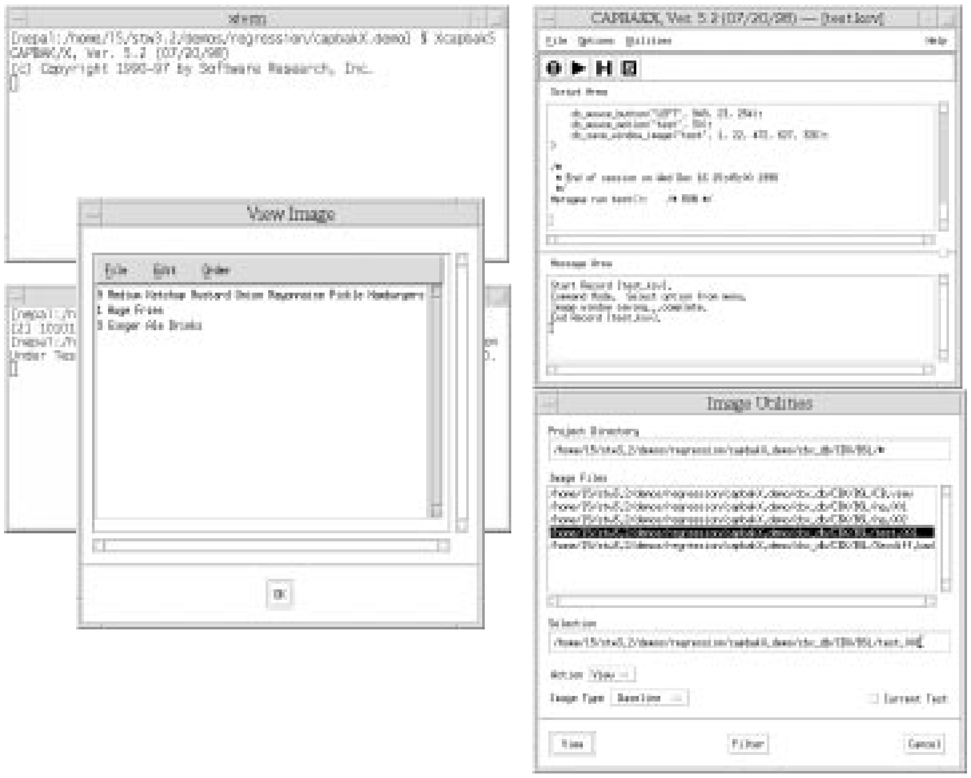
2.1.10 Viewing a Captured Image

To display the captured baseline image:

1. Select `test.001` by double-clicking it in the **Files** list box. You can also highlight or type in the file name and then click on **OK** or press the **Enter** key.
2. Select **Baseline** from the **Image Type** menu.
3. Select **View** from the **Action** menu, and then click on the **View** button in the lower-left-hand corner (this button will toggle to whatever option is chosen in the **Action** menu).

   The saved mouse image you captured pops up.
4. When you are finished viewing the image, click **OK**, and it disappears.
5. To exit the **View Images** window, click **Done**.

When the saved window is viewed, the display should look like this:



```
[nepal1:/home/15/stw3.2/demos/regression/capbakX.demo] $ Xcapbak5
CAPBAK/X, Ver. 5.2 (07/20/98)
(c) Copyright 1990-97 by Software Research, Inc.
```

```
9 Medium Ketchup Mustard Onion Mayonnaise Pickle Hamburgers
1 Huge Fries
5 Ginger Ale Drinks
```

```
Start Record [test.korv].
Command Mode.  Select option from menu.
Image window saving.../complete.
End Record [test.korv].
```

FIGURE 14 Viewing a Saved Image

2.1.11 Playing Back the Keysave File

1. Click the Play button.

   ▶

   The *CAPBAK/X* window's status window reads:

   `Start Playback.`

   Mouse movements and application-under-test keystrokes play back exactly as you entered them during the recording.
2. **As the keysave file is playing back, it's best not to interfere with the test by moving the mouse or entering keystrokes.**
3. The Motifburger window appears. You will see it jump to the location it was in during recording. This is *CAPBAK/X*'s automatic output synchronization.
4. Watch the Motifburger window be captured. The image you captured during your recording is automatically re-captured as a response file during playback. This is indicated by status window message:

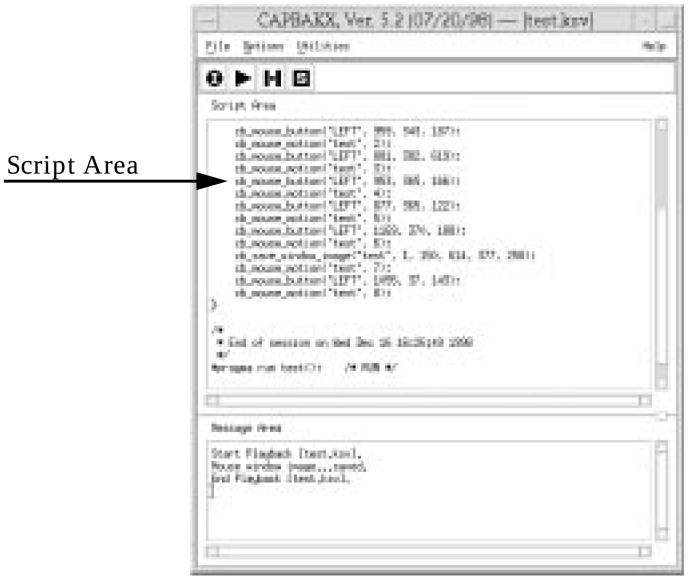
   `Mouse window image saving...complete.`

   Playback ends when the status window reads:

   `End Playback.`
5. Close the Motifburger window by clicking the mouse button on the upper-left-hand button of the window. Drag the mouse to **Close** and let go of the mouse button.

Note: If your keystrokes and / or mouse clicks don't play back and you are in OpenWindows, you did not invoke your window session with the **`-defeateventsecurity`** switch.

2.1.11.1 *CAPBAK/X* Script Area During Recording and Playback



```
    cb_mouse_button("LEFT", 999, 548, 137);
    cb_mouse_motion("test", 2);
    cb_mouse_button("LEFT", 881, 392, 615);
    cb_mouse_motion("test", 3);
    cb_mouse_button("LEFT", 853, 385, 166);
    cb_mouse_motion("test", 4);
    cb_mouse_button("LEFT", 877, 508, 122);
    cb_mouse_motion("test", 5);
    cb_mouse_button("LEFT", 1369, 376, 188);
    cb_mouse_motion("test", 6);
    cb_save_window_image("test", 1, 350, 614, 377, 288);
    cb_mouse_motion("test", 7);
    cb_mouse_button("LEFT", 1499, 37, 145);
    cb_mouse_motion("test", 8);
}

/*
 * End of session on Wed Dec 16 16:26:49 1998
 */
#pragma run test();    /* RUN */
```

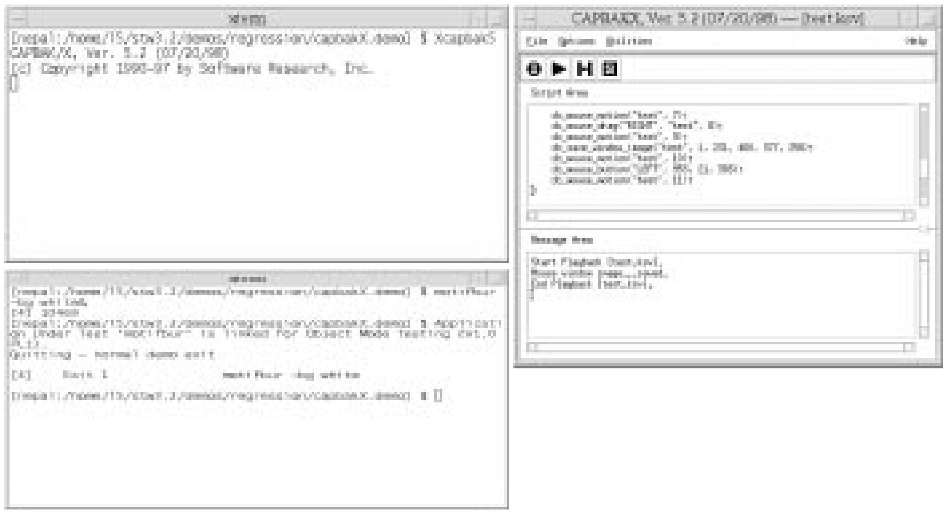
```
Start Playback [test.ksv].
Mouse window image...saved.
End Playback [test.ksv].
```

FIGURE 15 CAPBAK/X Script Area During Recording

Keeping the **Script Area** of the *CAPBAK/X* main window visible during recording and playback allows you to see, line-by-line, the events that are being recorded in the keysave file.

The advantage to the **Script Area** being displayed during recording is that scripts can be edited if a mistake is made. For example, if you forgot to do something during your test, you can click the pointer on the line where it was omitted. Then you can either type in the line of the test you want there (see Appendix B of the User’s Guide, “*CAPBAK/X* Function Calls”) or return to the GUI and click on the appropriate buttons.

When a playback is ending, the display should look like this:


xterm
CAPBAK/X, Ver. 5.2 (07/20/98) — [test.ksv]
File Options Utilities Help
Script Area
Message Area

```
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $ Xcapbak5
CAPBAK/X, Ver. 5.2 (07/20/98)
(c) Copyright 1990-97 by Software Research, Inc.
[]
```

```
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $ motifbur
-bg white&
[4] 23409
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $ Applicati
on Under Test 'motifbur' is linked for Object Mode testing (v1.0
[illegible]
Quitting - normal demo exit

[4]    Exit 1                motifbur -bg white

[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $ []
```

```
        [illegible]
}
```

```
Start Playback [test.ksv],
[illegible]
End Playback [test.ksv],
```

FIGURE 16 Ending a Playback Session

2.1.12 Reviewing Test Results

When you have completed the preceding steps (making sure to use the **Save** option from the **File** pull-down menu after recording), you should have three files saved to three different directories:

- A keysave file named `test.ksv`. This is saved in the current working directory.
- A baseline image file named `test.001`. This is the saved image you captured during the recording, and it is saved in the `cbx_db/CBX/BSL` sub-directory.
- A response image file named `test.001`. This is the saved image re-captured during the playback, and it is saved in the `cbx_db/CBX/RSP` sub-directory.

The image files (baseline and response) should be identical. You can view these files using the **View Images** window.

1. Click on the *CAPBAK/X* window's **Utilities** menu and select the **Image Utilities** submenu.
2. Select the baseline image file `BSL/test.001` and **View** it. Then do the same with the response image file `RSP/test.001` (as in the previous steps, you select the same test name for both, but use the **Image Type** menu to select the kind of image to view).
3. Using **Diff** will tell you if there are differences between the files; it will not display the differences. By displaying them one after the other you can make visual comparisons between the baseline and the response images.

2.1.13 STEP 13: Signoff

To complete this session:

1. Click inside the displayed images and they should disappear.
2. Click on the **View Images** window's **Done** button to exit.
3. Click on the *CAPBAK/X* window's **File** menu and select **Exit**.

2.2 Making Object Mode and Mixed-Mode Recordings

In ObjectMode testing, the application’s widgets are directly activated during test playbacks, so that their precise X and Y coordinates are not crucial to the success of the test. Because of this, ObjectMode is robustly portable across hardware and operating system platforms. This section describes how to perform ObjectMode and mixed mode recording by switching from TrueTime to ObjectMode.

2.2.1 Using Motifburger as a Sample Application Under Test

The demonstration application-under-test (AUT) provided for Object-Mode testing is **Motifburger**. The files for Motifburger are located in:

`$SR/demos/regression/capbakX.demo`.

For more information on these files, open the `README` file in this directory.

For a description of how Motifburger works, (See Section 2.1.4 - “Meet Motifburger” on page 15.).

Initialize two xterm-type windows, one to serve as your AUT window and the other to serve as your ***CAPBAK/X*** invocation window.

2.2.2 Invoking Motifburger for ObjectMode Testing

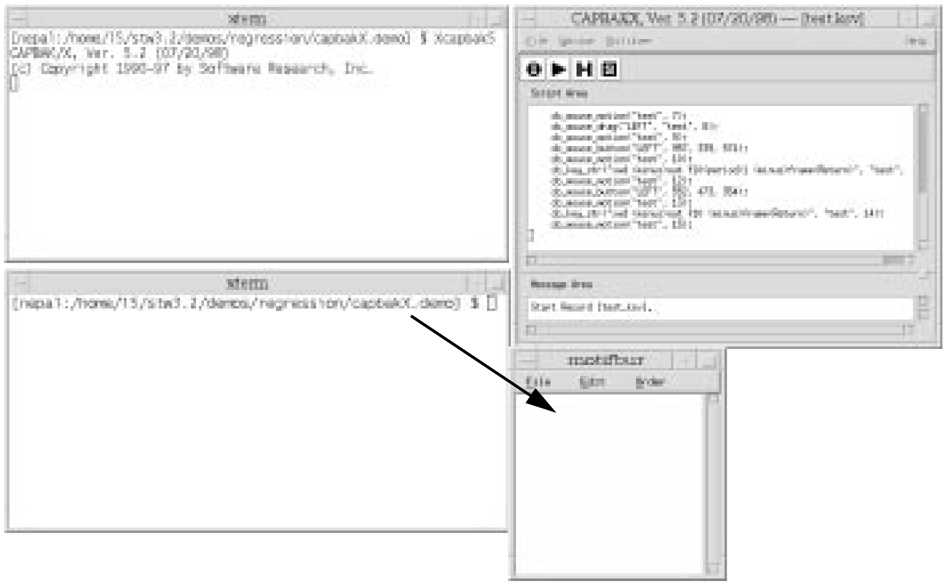
To invoke the sample AUT, type in the AUT window:

```
motifbur
```

The following message should appear:

```
Application Under Test ‘motifburger’ is linked
for widget-level testing
```

The **Motifburger** window will pop up.



```
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $ Xcapbak5
CAPBAK/X, Ver. 5.2 (07/20/98)
(c) Copyright 1990-97 by Software Research, Inc.
```

```
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $
```

```
Start Record [test.ksv].
```

FIGURE 17 Invoking Widget-Aware AUT

2.2.3 ObjectMode Recording from the *CAPBAK/X* Window

2.2.3.1 Invoking Motifburger

It is necessary to invoke the widget-aware AUT before starting Object-Mode recording. If you do not invoke the widget-aware application, the following lines will appear in the message area if you try to start an ObjectMode recording:

```
Unable to find CAPBAK-aware AUT. Is AUT up?
Not in Widget mode
```

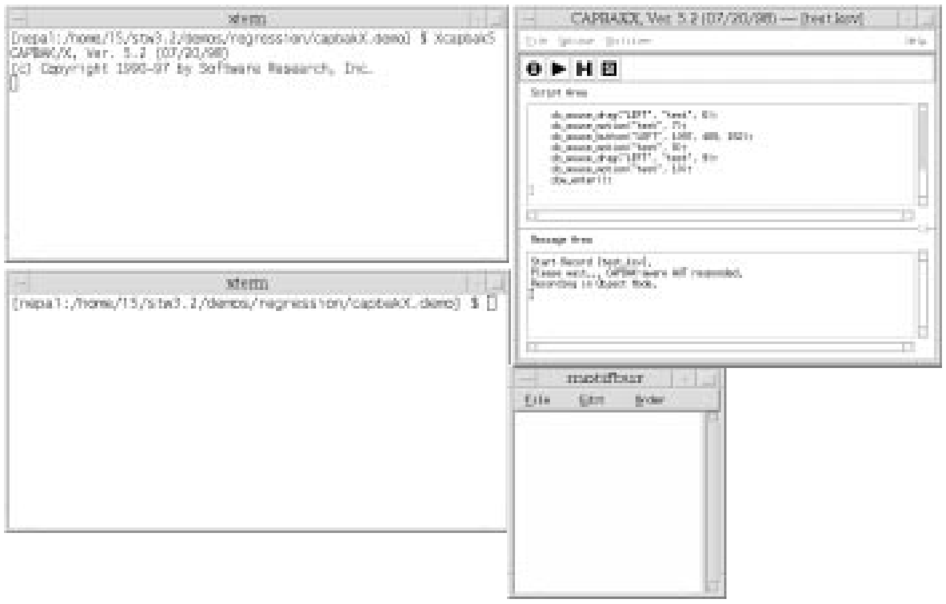
If this happens, re-invoke **motifbur** (the AUT). Make sure that the **motifbur** main window and the *CAPBAK/X* main window are open. The message area will read:

```
Please wait...AUT responded.
```

2.2.3.2 Beginning Recording in TrueTime Mode

To record in ObjectMode, begin recording in TrueTime (see preceding section), and use **CTRL+F2** to enter ObjectMode. Be sure to depress and release the **F2** key while the **CTRL** key is depressed.

You can initiate ObjectMode without starting in TrueTime by modifying the SR resource files. See Chapter Five in the *CAPBAK/X* User's Guide, "Setup for ObjectMode Recording," for more information.



```
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $ Xcapbak5
CAPBAK/X, Ver. 5.2 (07/20/98)
(c) Copyright 1990-97 by Software Research, Inc.
```

```
[nepal:/home/15/stw3.2/demos/regression/capbakX.demo] $
```

```
[illegible]
```

```
Start Record [test.kov].
Please wait..., CAPBAK/aware AUT responded.
Recording in Object Mode.
```

FIGURE 18 Switching to ObjectMode

2.2.3.3 Creating a Test

Click the recording button **(I)**. The message will change to `Recording`. *CAPBAK/X* will record all ensuing GUI activities.

Create a recorded session using the AUT using the same procedure as in TrueTime mode, described in Section 1.6.

2.2.3.4 Capturing a Window Image

You can capture images during the ObjectMode test using the same procedure as for TrueTime, described in Section 1.7.

2.2.3.5 Completing an ObjectMode Recording Session

Exit ObjectMode recording by pressing the **CTRL** + **F2** keys simultaneously. The *CAPBAK/X* message area will read:

```
Exiting Widget Mode
```

You can now return to TrueTime recording or simply end the session. You can end the session by using the **F9** function key or clicking the mouse on the **S** (Stop) button.

2.2.3.6 Viewing Images Captured During the Test

To see the images you captured during your ObjectMode test recording, follow the same steps as for TrueTime, as described in Sections 1.9 and 1.10.

2.2.3.7 Playing Back the Keysave File

Again, the procedure is the same as for ObjectMode: press the playback icon. See Section 1.11.

2.2.3.8 Reviewing Test Results

As in TrueTime recording, if you captured an image during your Object-Mode test session, this baseline image has been saved to a file named `test.001` in the `cbx_db/CBX/BSL` directory.

The keysave file containing the replayable recording of your entire test session is saved in the current directory as `test.ksv`.

If you played back the test, and the test contained a the procedure of saving an image, the image saved during playback is saved as `test.001` in the `cbx_db/CBX/RSP` directory.

2.2.4 ObjectMode Recording Summary

The above section on ObjectMode recording describes the most convenient way to use the ObjectMode capability of *CAPBAK/X*: by toggling over into ObjectMode using **CTRL+F2**. Starting up in ObectMode requires changes to the SR resource files.

2.3 Summary

This chapter is an introduction to the procedures for using *CAPBAK/X*'s TrueTime and ObjectMode to record test sessions, capture images, view captured images, play back tests, and compare baseline to response images.

CHAPTER 3

CAPBAK/X Graphical User Interface

INFORMATION: Summary of *CAPBAK/X* windows, menus and commands.
SEE ALSO: Individual application of commands is described in Chapter 4.

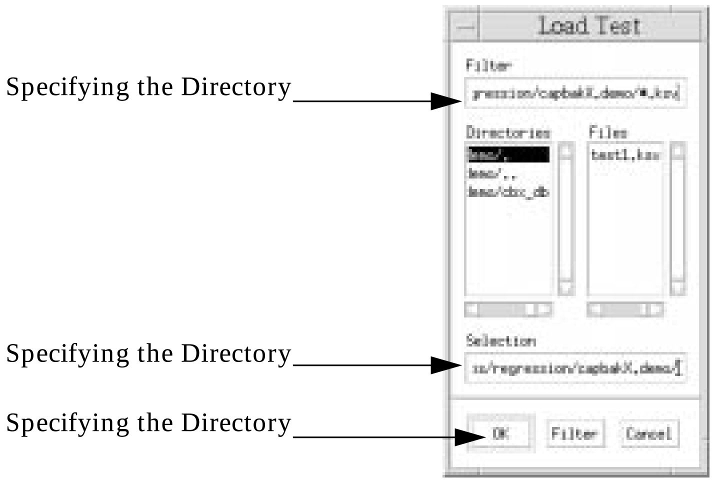
LEVEL: This chapter is intended for beginning to intermediate-level users.

3.1 Basic OSF/Motif User Interface

This section demonstrates the use of file selection dialog boxes, help menus, message dialog boxes, option menus, and pull-down menus. If you are familiar with the basic OSF / Motif graphical user interface (GUI) style, you can go on to another section (See Section 3.2 - "The CAPBAK / X Window" on page 46.).

3.1.1 File Selection Windows

CAPBAK/X file selection windows allow you to select an existing test file name or specify a new test file name. The **Load Test** window (below) will give you a list of existing keysave files; the **New Test** window will allow you to name a new keysave file.



Filter: gression/capbakX_demo/*.ksv

Directories: demo/. demo/.. demo/cbx_db

Files: test1.ksv

Selection: ss/regression/capbakX_demo/

FIGURE 19 File Selection Window

The components of the file selection window are as follows:

Filter entry box	Specifies a directory mask. When the **Filter** button is clicked, the file selection window displays only files or directories that match the indicated mask (or pattern).
Directories list box	Lists directories in path defined in the **Filter** entry box.
Files list box	Lists files for the path defined in the **Filter** entry box.
Scroll bars	Allows vertical and horizontal movement within the **Directories** and **Files** list boxes.
Selection entry box	Accepts a file name for selection.

Use the three push buttons at the bottom of the window to issue commands:

OK	Accepts the file in the **Selection** entry box as the new file or the file to be opened and then exits the window.
Filter	Applies the pattern specified in the **Filter** entry box and lists the directories and files that match that pattern.
Cancel	Cancels any entered selections and exits the window.
Scroll bars	Move up / down and side / side in the **Directories** and **Files** list boxes.

3.1.2 Using a File Selection Window

The file selection operation can be restricted to a named region (directory path) by either:

- Typing a directory path name in the **Filter** entry box, or
- By clicking on a path name in the **Directories** list box.
- The **Filter** push button is then clicked.

To select a file name, perform one of the following activities:

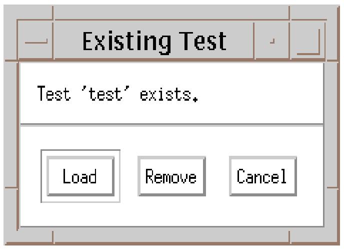
- double-click on the file in the **Files** list box.
- Highlight the file in the **Files** list box or type in the file name in the **Selection** entry box and click on **OK**.
- Highlight or type in the file name and press the **Enter** key.

When a procedure requires a file to be created, create a new file by either:

- Using the **New** option in the **File** pull-down menu, and entering a new test name in the appropriate dialog box, or
- Double-clicking on an existing file in the **Files** list box to be over-written.

3.1.3 Selecting an Already-Existing Test

If you choose a file name that already exists, *CAPBAK/X* will provide you with a “warning” pop-up box which allows you to insert or append information to the existing keysave file (**Load**), remove the existing test and record a new test with the same name (**Remove**), or cancel the procedure (**Cancel**).



Test 'test' exists.

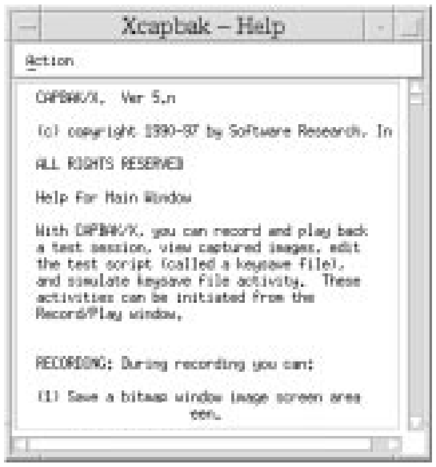
FIGURE 20 Existing Test Dialog Box

3.1.4 Help Windows

CAPBAK/X provides on-line information for each of its windows. When a window's **Help** option is activated, an information window containing pertinent text is displayed. In other words, if invoked from the *CAPBAK/X* window, the **Help** window will automatically display information relevant to the *CAPBAK/X* window.

To access on-line help:

1. Click on the window's **Help** option. The **Help** window pops up, displaying context-sensitive help.



```
CAPBAK/X,  Ver 5.n

(c) copyright 1990-97 by Software Research, In

ALL RIGHTS RESERVED

Help For Main Window

With CAPBAK/X, you can record and play back
a test session, view captured images, edit
the test script (called a keysave file),
and simulate keysave file activity.  These
activities can be initiated from the
Record/Play window.


RECORDING: During recording you can:

(1) Save a bitmap window image screen area
              een.
```

FIGURE 21 Help Window

The displayed text can be traversed by using either the vertical or horizontal scroll bars.

2. Click on the **Action** menu and select the **Search** option to search for specific text. The following dialog box pops up:

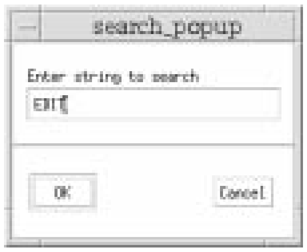
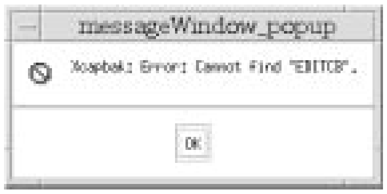


FIGURE 22 Search Dialog Box

3. Click the cursor in the **Enter string pattern to search** region and type in the desired text.
4. Either click on **OK** or press the **Enter** key.

   If the pattern is found, then the window will automatically scroll to the location of the specified pattern.

   If the pattern is *not* found, the following message box is displayed:



```
Xcapbak: Error: Cannot find "EDITCB".
```

FIGURE 23 Help Message Box

Note: If the **Help** window is currently displayed and a **Help** option from another window is activated, the **Help** window automatically scrolls to the relevant text for the new window.

5. To close the **Help** window, click on the **Action** menu and select the **Exit** option.

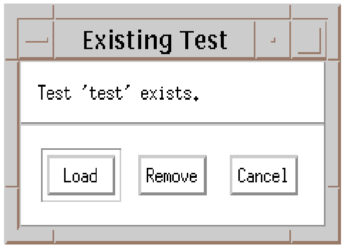
3.1.5 Message Boxes

Pop-up message boxes serve three purposes:

- Display warnings and error information.
- Prompt for a command.
- Request verification for command execution.

To remove a message box or to execute the current command, click on the **OK** button.

To cancel a command, click on the **Cancel** button.



Test 'test' exists.

FIGURE 24 Message Box

3.1.6 Option Menus

An option menu includes selections from a list. Usually, only the default menu option is visible.

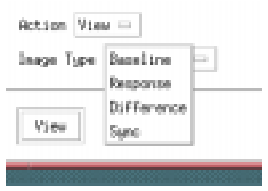


FIGURE 25 Option Menu (Image Type Options)

To use an option menu, follow these steps:

1. Click on the option menu.
2. After clicking on the menu, the list of choices is visible.
3. Drag the mouse to the menu option you want.
4. Let go of the mouse.
5. The selected menu option should now be visible, indicating that it is activated.

3.1.7 Pull-Down Menus

Pull-down menus are located within the menu bar of *CAPBAK/X* windows. They often contain several options.

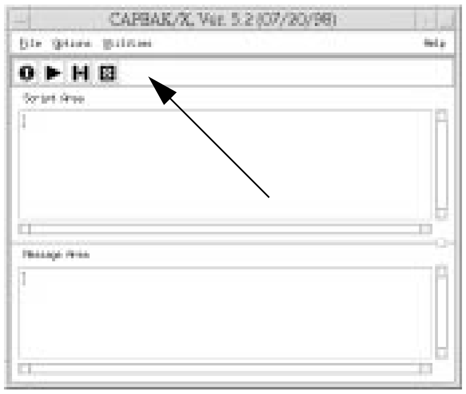


FIGURE 26 Pull-Down Menu

To use pull-down menus and their options, perform the following steps:

1. In the menu bar, place the mouse pointer over the menu name.
2. Display the menu's options by holding the left mouse button down.
3. While holding down the left mouse button, slide the mouse pointer to the desired menu option. The menu option is highlighted in reverse shadow.

Note: Three dots to the right of a menu item indicates that selecting the item will display a pop-up window, such as a file selection window. An arrow to the right of the menu indicates that item has a submenu (or cascading menu).

4. To activate a command, release the mouse button while the desired item is highlighted. To exit without selecting an item, simply drag the mouse pointer off the menu before releasing the mouse button.
5. To display the submenu, slide the mouse pointer over the arrow. You can then select an item on the submenu.
6. Release the mouse button while the desired item is highlighted to activate the command. To exit without selecting anything, simply drag the mouse pointer off the menu before releasing the mouse button.

3.2 The *CAPBAK/X* Window

The *CAPBAK/X* window displays all the commands and menus to operate *CAPBAK/X*, including capture/replay and the utilities to view captured images and get system information.

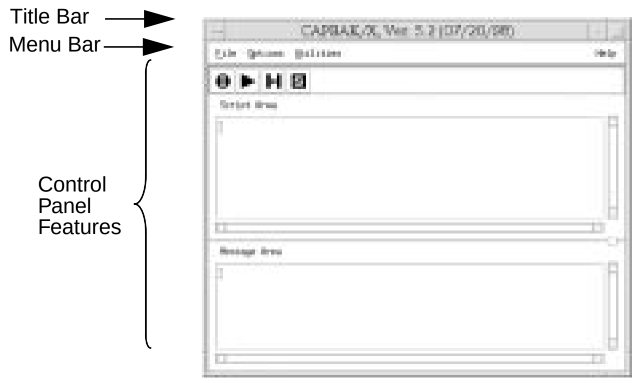


FIGURE 27 CAPBAK/X Window

The window is divided into the following parts:

1. Title bar.
2. Menu options (bar).
3. Control panel features.

3.2.1 Menu Bar

The title bar spans the length of the top of the *CAPBAK/X* window and contains the following items:

1. **File** menu.
2. **Options** menu.
3. **Utilities** menu.
4. **Help** option.

3.3 File Menu

The **File** menu allows you to open up new keysave files, load previously-recorded files, save files upon completion of recording, specify a directory to which your work will be saved, clear or save the information in the **Message Area**, and exit *CAPBAK/X*.

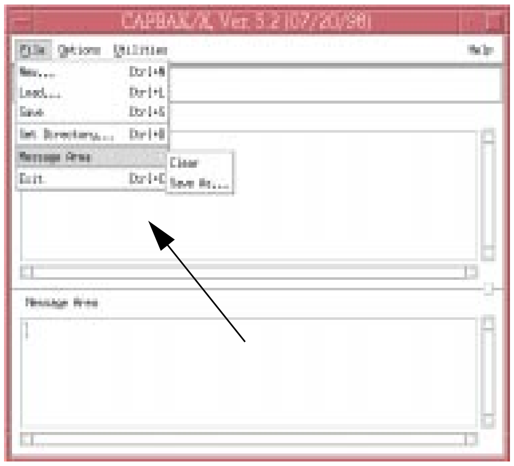


FIGURE 28 File Menu

New	Invokes a **new_popup** window. In this window you can assign a new test name or type in an existing one.
Load	Invokes a window displaying the available keysave files (`.ksv` files) in your current working directory that can be chosen for playback.
Save	After a recording session is complete, you must use this option if you want to save the file to your current working directory. It will automatically be given a .ksv extension (e.g. `test` will appear as `test.ksv` in your current working directory). If you want to save your file somewhere other than your current working directory, see **Set Directory** option.
Set Directory	Allows you to specify a directory into which your work will be saved, if you do not want it saved in the current working directory.
Message Area	Allows you to clear the current information in the **Message Area**, or save it to a file.
Exit	Closes *CAPBAK/X*.

3.4 Options Menu

The **Options** menu allows you to set record and playback options and change the function key defaults. Each option is described below.

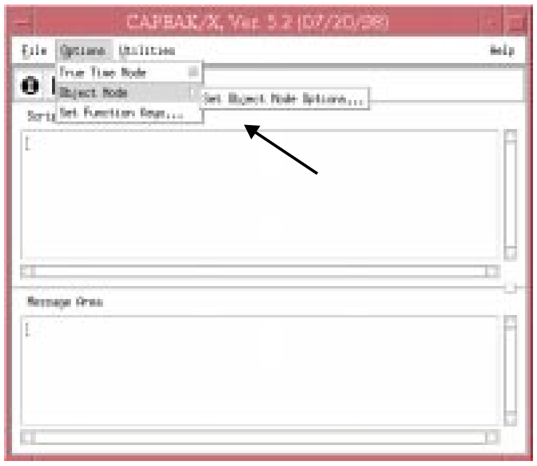


FIGURE 29 Options Menu

3.4.1 TrueTime Mode Options

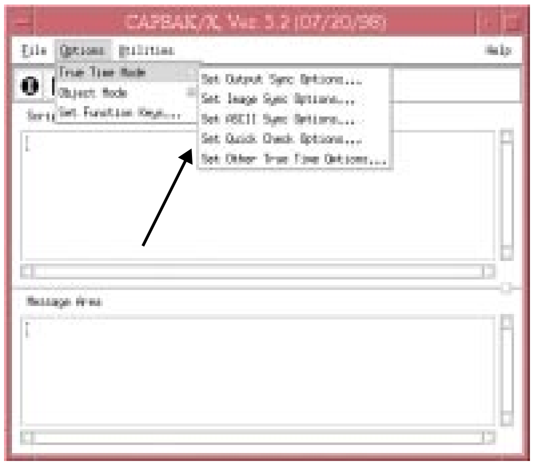
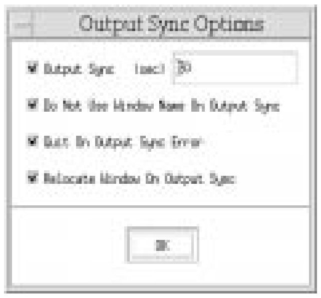

CAPBAK/X, Ver. 5.2 (07/20/98)
File
Options
Utilities
Help
True Time Mode
Object Mode
Set Function Keys...
Set Output Sync Options...
Set Image Sync Options...
Set ASCII Sync Options...
Set Quick Check Options...
Set Other True Time Options...
Message Area

FIGURE 30 TrueTime Options pull-down Menu

3.4.1.1 Set Output Sync Options

In the X Window System environment, windows are likely to come up at varying locations on screen each run. The **Output Sync Options** window's options synchronize on window pop ups. The options are all defaulted to on.



[✓] Output Sync (sec) 30
[✓] Do Not Use Window Name On Output Sync
[✓] Quit On Output Sync Error
[✓] Relocate Window On Output Sync

FIGURE 31 Output Sync Options Window

Output Sync (sec) Defines the maximum time-out period **(sec)** that output synchronization, which controls synchronization on window pop ups, will "wait" before continuing. The default is 30 seconds. If playback fails to synchronize in that time, playback continues unless the **Quit on Output Sync Error** option is turned on. During a recording session, this option places a `cb_sync_win_activate` statement in the keysave file, which playback interprets.

Warning: This feature may not work properly for certain applications that are not ICCCM compliant.

Do Not Use Window Name On Output Sync

Ignores a window's title name during output synchronization. This is useful when a window's name changes.

Quit On Output Sync Error

Terminates the playback session, if output synchronization fails.

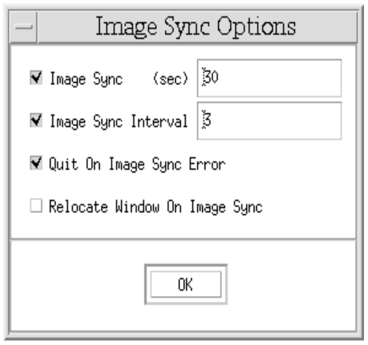
Relocate Window On Output Sync

If a window pops up in a different place during playback than it did during the recording session, this option moves the window to the right location. The **Output Sync** option must accompany this option.

Note: If you a record a session with these options on, you must also leave these options on during playback for output synchronization to work properly. If you record a session with these options off and then turn them on for playback, output synchronization will not work because the necessary `[cb_sync_win_image]` statement will not be recorded.

3.4.1.2 Set Image Sync Options

These options apply to the playback of a test session that used the **Hotkey** window's **Sync Image** button or the **F3** or **F4** function keys to define partial images, pull-down menus or windows for synchronization. *CAPBAK/X* holds back test execution until the image or window is redrawn or is found.



☑ Image Sync (sec) 30
☑ Image Sync Interval 3
☑ Quit On Image Sync Error
☐ Relocate Window On Image Sync

FIGURE 32 Image Sync Options Window

Image Sync (sec) Defines the time-out period (sec) that playback waits for a specified image or window to match a recording's drawn image. The default is 30 seconds. Unless the **Quit On Image Sync** is turned on, playback proceeds even if the image does not match.

Image Sync Interval

Defines the interval time for image synchronization to grab the image it is trying to match or find. The default interval is three seconds; thus, since **Image Sync (sec)** is set to 30, *CAPBAK/X* will search for an image 10 times before synchronization fails.

Quit On Image Sync Error

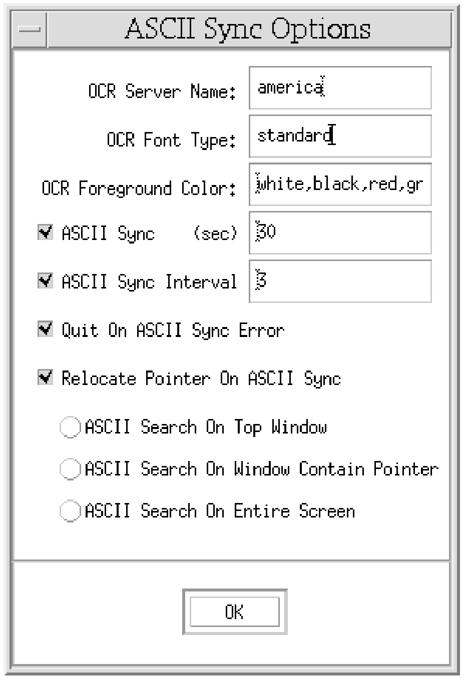
Terminates the playback session, if image synchronization fails.

Relocate Window On Image Sync

Determines if playback tries to find an image or match a baseline's image. If this option is on, playback tries to find the image on the screen that matches the saved baseline image in a specified time period and then moves that image to the correct location. If this option is off, playback waits for a specified image to match a baseline's image drawn image.

3.4.1.3 Set ASCII Sync Options

The **ASCII Sync Options** determine parameters for character recognition level recording.



OCR Server Name: america

OCR Font Type: standard

OCR Foreground Color: white,black,red,gr

[✓] ASCII Sync (sec) 30

[✓] ASCII Sync Interval 3

[✓] Quit On ASCII Sync Error

[✓] Relocate Pointer On ASCII Sync

◯ ASCII Search On Top Window

◯ ASCII Search On Window Contain Pointer

◯ ASCII Search On Entire Screen

FIGURE 33 ASCII Sync Options Window

OCR Server Name:

Specifies the name of the host machine where the OCR license server is installed. During the installation procedure, the host machine's default is automatically appended to the *SR* file. Please refer to Chapter 4 of the *Installation Instructions* for further information on appending the default to the *SR* file.

OCR Font Type: Specifies the name of the font family for character synchronization. During the installation procedure, the default font family is automatically placed in the *SR* file. Please refer to Chapter 4 of the *Installation*

Instructions for further information on appending the default to the *SR* file.

OCR Font Colors: If the OCR software cannot find the colors in the SR file, then it looks for type in the specified colors; up to four colors can be listed. The colors must be legal X-color names (see the *X User's Guide* for a description of legal color names). If there is more than one, then the color names must be separated by a comma, with no space in between them.

ASCII Sync: Defines the time-out period that playback waits for a specified character string to be found. The default is 30 seconds. Unless the **Quit On ASCII Sync Error** option is turned on, playback proceeds even if the character string is not found.

ASCII Sync Interval:

Defines the interval time for playback to search for the character string it is trying to find. The default interval is three seconds.

Quit On ASCII Sync:

Terminates the playback session, if character synchronization fails.

Relocate Pointer On ASCII Sync:

The option performs two functions:

If a character string moved and is found, this option moves the mouse pointer to the new location.

Changes where the search for the character string is done in respect to the screen.

Note: If this option is not selected, playback restricts the area of the character string search to the area of the rectangle captured. If this option is turned on and none of the below sub-options are selected, the location of the search is defaulted to the top window, or "focus" window.

ASCII Search On Top Window

Searches for a captured character string in the screen's topmost window, sometimes referred to as the "focus" window.

ASCII Search On Window Contain Pointer

Searches for a captured character string in the window that contains the mouse pointer.

ASCII Search On Entire Screen

Searches for a captured character string throughout the entire screen. This may take the system a long time.

3.4.1.4 Quick Check Options Window

The **Quick Check Options** window's options affect how comparisons are done during playback. It compares for differences between playback's actual results with the recording session's expected results. The following options are all defaulted to off.

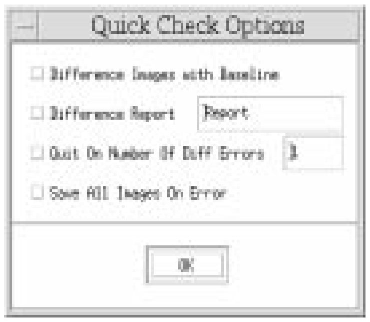


FIGURE 34 Quick Check Options Window

Difference Images with Baseline

Automatically compares response images (images captured during playback) to their corresponding baseline images (images captured during the initial recording) during playback.

This option must be turned on in order for the other *Quick Check* options to work.

Difference Report

Creates a pass/fail results report of expected image and actual image comparisons (defaulted to *Report*).

Quit on Number Of Diff Errors

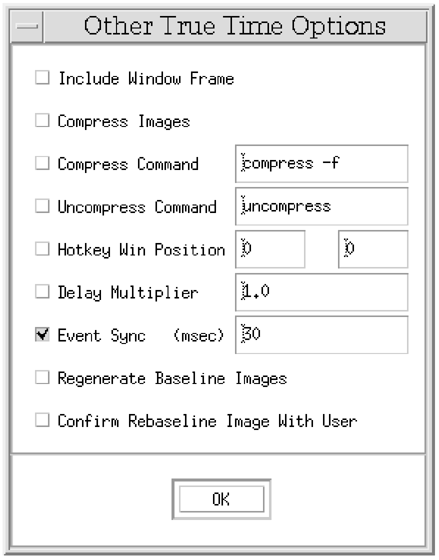
Terminates playback if the number of times a response image fails to match a baseline image reaches the number specified. If this option is not turned on, playback proceeds even when comparisons fail.

Save All Images on Error

If images fail to compare during playback, this option will save the response file (actual result) image. If this option is not turned on and there is a difference, no response file will be created.

3.4.1.5 Set Other TrueTime Options

This window sets various record / playback features.



Other True Time Options

☐ Include Window Frame

☐ Compress Images

☐ Compress Command: compress -f

☐ Uncompress Command: uncompress

☐ Hotkey Win Position: 0 0

☐ Delay Multiplier: 1.0

☑ Event Sync (msec): 30

☐ Regenerate Baseline Images

☐ Confirm Rebaseline Image With User

FIGURE 35 Other Options Window

Include Window Frame

Includes a window's frame when a window is captured. Default = Off.

Compress Images

Compresses captured images into files named *base-name.xxx.Z,* where *xxx* is the number sequence in which the image was captured, and *Z* indicates compression. This option is available for both the record and playback phases. Default = Off.

Compress Command

Allows you to change the command to compress any window or partial image you may save. The default command is set to `compress -f`. Default = Off.

Uncompress Command

Allows you to change the command to restore the compressed images when you view them.The default command is set to `uncompress`. Default = Off.

Hotkey Win Position

Allows you to set the X,Y coordinates of where you would like the **Hotkey** window to pop during a recording session. The **Hotkey** window allows you to do such things as place comments in the keysave file, save images, or terminate a test. You must turn this switch on if you want a different pop-up location for the window. The default is set to 0,0.

Delay Multiplier Allows you to quicken or slow the pace of playback. The higher the number, the slower playback is; the lower the number, the faster playback is. If the delay multiplier is set to a very fast speed, synchronization may be sacrificed. Every system varies so you should experiment with speeds. You must turn this option on if you want a different speed to apply to your test.The default is set to 1.0, which represents real time.

Event Sync (msecs) This option defines the maximum time that event synchronization, which controls synchronization on pull-down menus, will "wait" before continuing. The default is 30 milliseconds. If playback fails to synchronize in that time, playback continues. Default = On.

Regenerate Baseline Images

Saves images captured during playback to baseline image files instead of response image files. This feature can be very effective if you are transferring a test from one system to another system with different display characteristics, such as a color monitor to a monochrome monitor. This option will overwrite *all* baseline images.

Warning: In order for this option to work, all the **Quick Check** options and the **Confirm Images on Play Back** option must be turned off.

Confirm Rebaseline Image With User

During an ordinary playback session, all re-captured images are saved as response files. With this option turned on, a **Verify Baseline** window will pop up whenever playback encounters a saved partial image, window, or full-screen image:

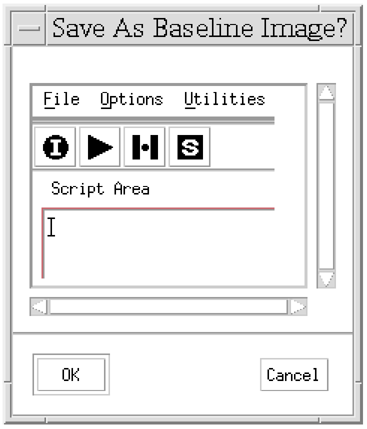


FIGURE 36 Verify Baseline Window

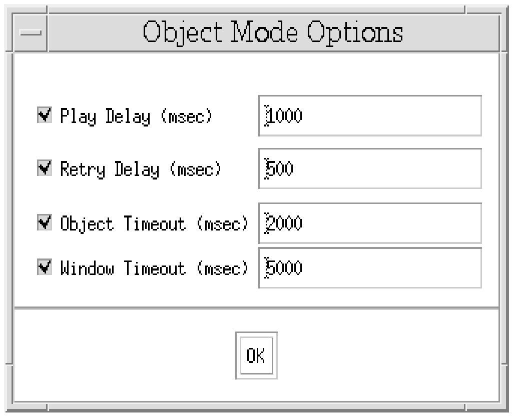
This window displays the captured image in its display area. You can use the scroll bars to move up / down and side / side for images that do not fit the size of the display.

Click on **YES** if you want the image to be saved as a baseline file; click on **NO** if you want it to be saved as a response file. Default = Off.

Warning: In order for this option to work, all the **Quick Check Options**, as well as the **Regenerate Baseline Images** option, must be turned off.

3.4.2 Set ObjectMode Options

The **Options** menu allows you to set record and playback options (True-Time and ObjectMode) and change the function key defaults. (See Figure 29 on page 48.)



☑ Play Delay (msec) 1000
☑ Retry Delay (msec) 500
☑ Object Timeout (msec) 2000
☑ Window Timeout (msec) 5000

FIGURE 37 ObjectMode Options Window

Play Delay

This specifies the number of milliseconds delay between events. Default is set to 1 second. You can set this option to 0 for fastest possible playback.

Retry Delay

This option specifies the amount of time in between *CAPBAK/X* attempts to locate the object on the screen during playback. The default value is .5 seconds, meaning *CAPBAK/X* will wait that amount of time between searches.

Object Timeout

This option specifies the amount of time *CAPBAK/X* will wait for an object to appear onscreen during playback. Default is 2 seconds. Playback will continue after this amount of time.

Window Timeout

This option specifies the amount of time *CAPBAK/X* will wait for a top-level window during playback. Default is 5 seconds. Playback is terminated if window is not found.

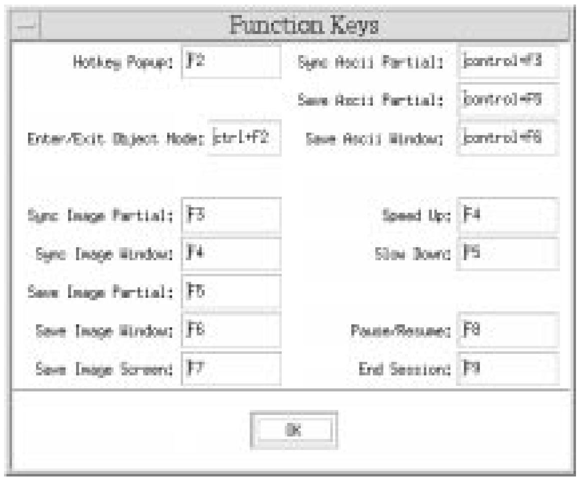
Note: Changes made to any options are not saved between invocations of *CAPBAK/X*. To make permanent changes to options, please refer to Appendix A, “CUSTOMIZING *CAPBAK/X*”, for instructions on editing the resource file *SR*.

3.4.3 Set Function Keys

Several of *CAPBAK/X* recording and playback commands can be activated using keyboard functions keys. During a recording session, you can use these keys to save images or ASCII text, define images or ASCII text for synchronization, enter and exit ObjectMode recording, pause, resume and terminate the test session.

During playback, the function keys allow you to speed up or slow down. You can also safely pause the playback session, resume that session, or terminate a session altogether. You can re-configure the function keys using the **Function Keys** window, or by editing the appropriate lines in the SR file**.**

Note: The use of any function-key sequence must be very precise. If **CTRL** or **ALT** is part of the sequence, these keys must be held down the *entire time* while the function key is pressed and then let go.



Hotkey Popup:	F2	Sync Ascii Partial:	control+F3
		Save Ascii Partial:	control+F5
Enter/Exit Object Mode:	ctrl+F2	Save Ascii Window:	control+F6
Sync Image Partial:	F3	Speed Up:	F4
Sync Image Window:	F4	Slow Down:	F5
Save Image Partial:	F5		
Save Image Window:	F6	Pause/Resume:	F8
Save Image Screen:	F7	End Session:	F9

FIGURE 38 Function Keys Window

3.5 Utilities Menu

The **Utilities** menu invokes the *CAPBAK/X* supplemental utilities, which let you view captured images and get system information.

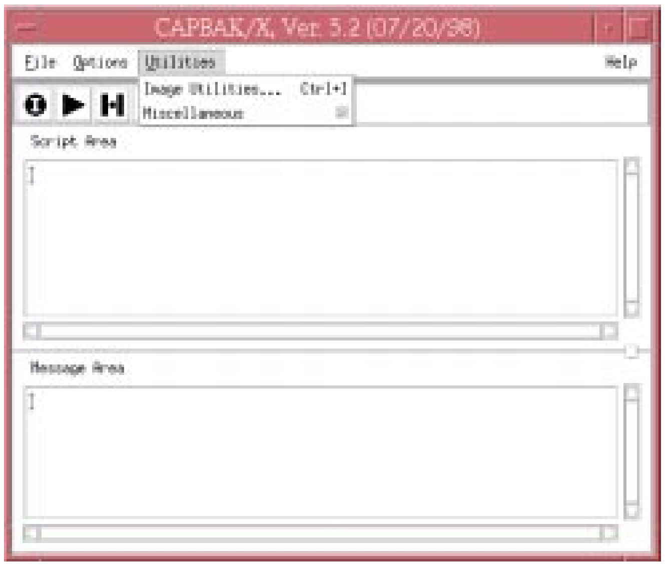
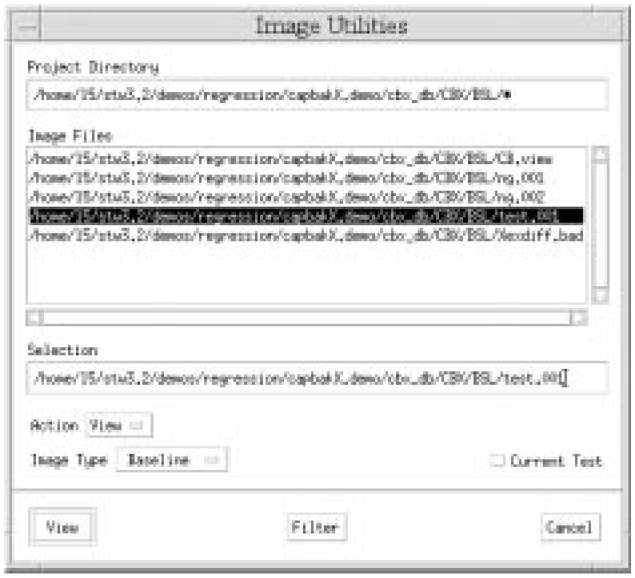


FIGURE 39 Utilities Menu

3.5.1 Image Utilities

Invokes the **View Images** window, which allows you to view captured images. The previous chapter gives a review of commands (See Section 2.1.10 - "Viewing a Captured Image" on page 26.).

3.5.1.1 View Images Window



Project Directory
/home/IS/stw3.2/demos/regression/capbakX_demo/cbx_db/CBX/BSL/*

Image Files
/home/IS/stw3.2/demos/regression/capbakX_demo/cbx_db/CBX/BSL/CB.view
/home/IS/stw3.2/demos/regression/capbakX_demo/cbx_db/CBX/BSL/ng.001
/home/IS/stw3.2/demos/regression/capbakX_demo/cbx_db/CBX/BSL/ng.002
/home/IS/stw3.2/demos/regression/capbakX_demo/cbx_db/CBX/BSL/test.001
/home/IS/stw3.2/demos/regression/capbakX_demo/cbx_db/CBX/BSL/Xexdiff.bad

Selection
/home/IS/stw3.2/demos/regression/capbakX_demo/cbx_db/CBX/BSL/test.001

Action: View

Image Type: Baseline

☐ Current Test

FIGURE 40 View Images Window

All functions for viewing captured images, full screen (root), full or partial windows are accessible from this window (accessible from the **Utilities** pull-down menu).

During your recording session full screen, window, and partial screen captures are automatically saved as *basename.xxx* (*xxx* represents the three-digit numeral assigned to each image, beginning with *basename.001*). During your playback session images are also saved as *basename.xxx*, but the are saved to the RSP directory instead of the BSL directory. Images saved for synchronization during playback are saved as *basename.xxx* <u>during the recording session only</u>, and must be matched during playback. These are saved to the SYN directory.

Images captured during your recording session and images re-captured during your playback session should be identical. If they are not, your playback test session failed. For this reason, **View Images** is a quick way to determine if your test passed or failed.

To compare two images, highlight the filename in the **Image Files** area. Then, choose **DIFF** from the **Image Type** menu in the bottom-left corner of the window. If you want to simply see a baseline, response or sync image, select the filename in the **Image Files** area, select the kind of file you want to view from the **Image Type** menu, and click on the **VIEW** button in the bottom-left corner of the window.

If you want to find a particular image file, select **FIND** from the **Action** menu.The button in the extreme bottom-left corner of the screen will toggle to whatever item you have chosen in the **Action** menu: **VIEW**, **DIFF** or **FIND**.

If during your recording and playback session you compressed all saved images, these images will be saved as *basename.xxx.Z*, where *xxx* represents the original sequence of captures and *Z* represents a compressed image.

You must turn on the **Other Options** window's **Compress Images** option in order for the **View Images** window to automatically decompress any compressed images for viewing purposes.

3.5.2 Miscellaneous

This pull-down menu allows you to get various system information, such as the X,Y offsets for your window manager, coordinates for all the windows onscreen or just the focus window, etc.

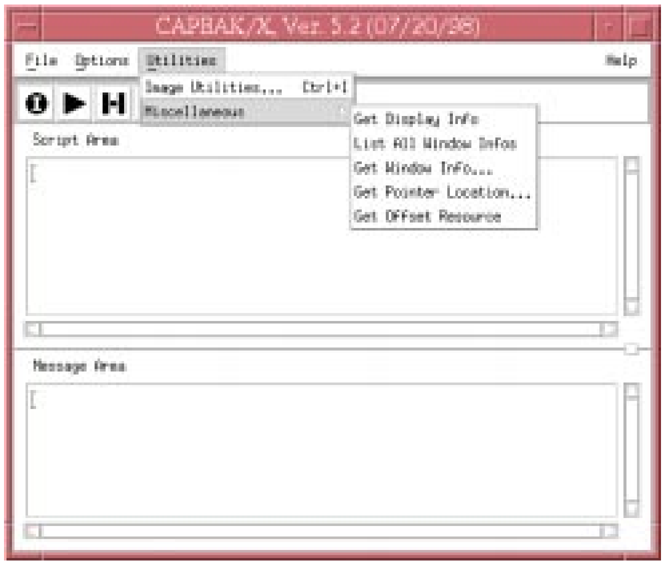
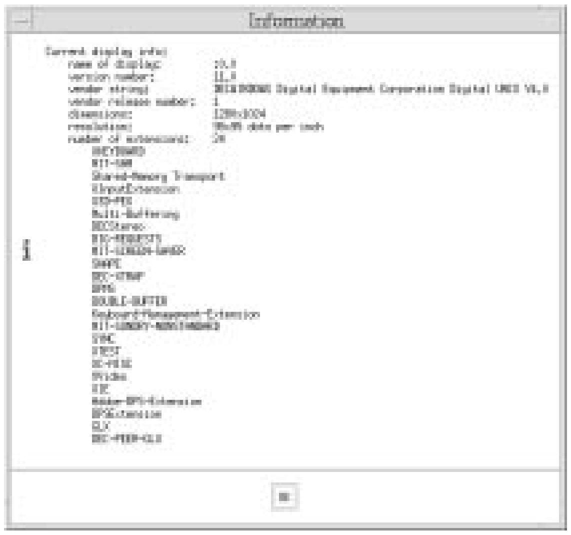


FIGURE 41 Miscellaneous Pull-Down Menu

3.5.2.1 Get Display Info

This pop-up dialog box tells you information about your current display, such as screen size, resolution, X-extensions, etc.

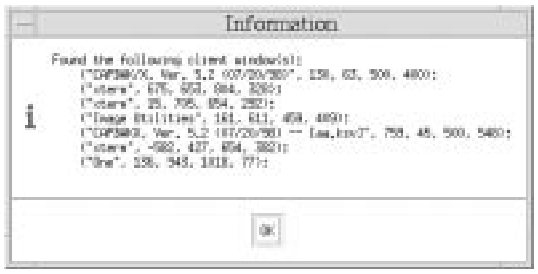


```
Current display info:
    name of display:          :0.0
    version number:           11.0
    vendor string:            DECWINDOWS Digital Equipment Corporation Digital UNIX V4.0
    vendor release number:    1
    dimensions:               1280x1024
    resolution:               95x95 dots per inch
    number of extensions:     24
        XKEYBOARD
        MIT-SHM
        Shared-Memory Transport
        XInputExtension
        X3D-PEX
        Multi-Buffering
        DECStereo
        BIG-REQUESTS
        MIT-SCREEN-SAVER
        SHAPE
        DEC-XTRAP
        DPMS
        DOUBLE-BUFFER
        Keyboard-Management-Extension
        MIT-SUNDRY-NONSTANDARD
        SYNC
        XTEST
        XC-MISC
        XVideo
        XIE
        Adobe-DPS-Extension
        DPSExtension
        GLX
        DEC-PEER-GLX
```

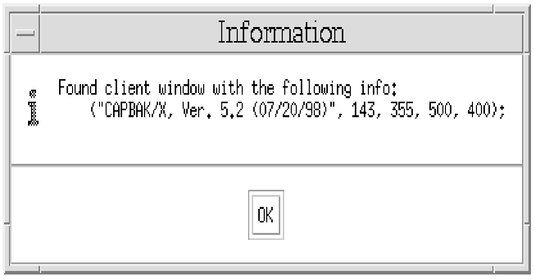
FIGURE 42 Get Display Info Window

3.5.2.2 List All Windows/ Get Window Info

This window gives information about all the windows currently displayed on your screen (**List All Windows**) or the focus window only (**Get Window Info**). **Get Window Info** will display the information from the window that is clicked on with the cross-hairs symbol; these cross-hairs appear after this option is chosen.



```
Found the following client window(s):
    ("CAPBAK/X, Ver. 5.2 (07/20/98)", 138, 63, 500, 400);
    ("xterm", 675, 653, 804, 328);
    ("xterm", 35, 705, 854, 282);
    ("Image Utilities", 161, 611, 458, 489);
    ("CAPBAK/X, Ver. 5.2 (07/20/98) -- [illegible]", 758, 48, 500, 540);
    ("xterm", -402, 427, 854, 382);
    ("Bnw", 138, 948, 1018, 77);
```

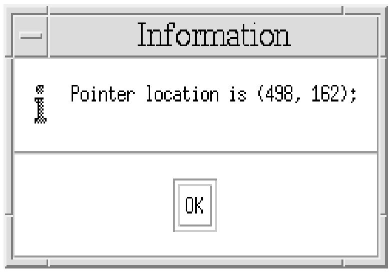


```
Found client window with the following info:
    ("CAPBAK/X, Ver. 5.2 (07/20/98)", 143, 355, 500, 400);
```

FIGURE 43 List All Windows (top), Get Window Info (bottom)

3.5.2.3 Get Pointer Location

When you choose this option, a cross-hairs symbol appears onscreen. This window will display the screen coordinates for whatever point you click onscreen.

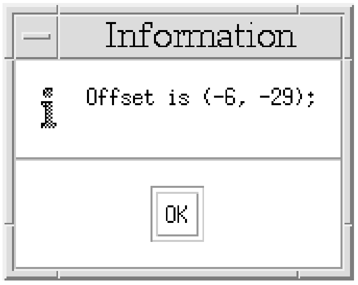


Pointer location is (498, 162);

FIGURE 44 Get Pointer Location Window

3.5.2.4 Get Offset Resource

This window displays the current x,y offset defaults for your window manager. It may be necessary to include this information in your SR file to ensure the proper operation of *CAPBAK/X*.



```
Offset is (-6, -29);
```

3.5.3 Help Option

The **Help** option invokes a **Help** window that describes the main features of the *CAPBAK/X* window.

3.5.4 Control Panel Features

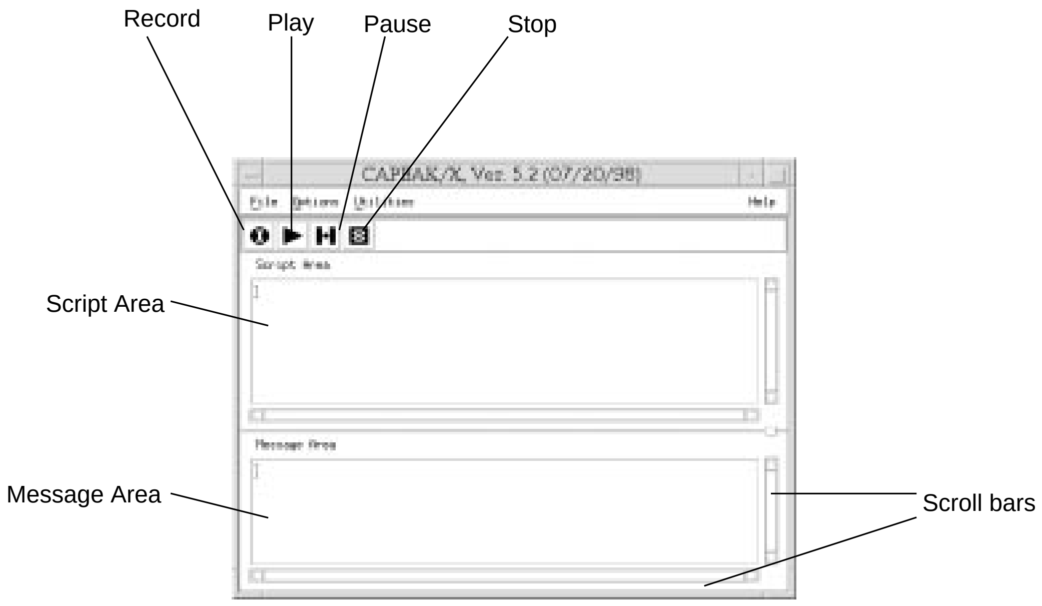


FIGURE 45 Control Panel Features

The window has the following control panel features:

Record button	Starts a recording session.
Play button	This button allows you to start a play back session.
Pause button	This button allows you to pause a recording session. Because you do not have control over the mouse during playback, you cannot use this button during a playback session. Instead, pause with the **F8** function key.
Stop button	This button allows you to stop a recording session. Because you do not have control over the mouse during playback, you cannot use this button during a playback session. Instead, use the **F9** function key.
Script area	Shows all the actions undertaken during a recording session.
Message area	Shows all the actions undertaken during a recording session.
Scroll bars	Allow you to maneuver inside the status window.

CHAPTER 4

Invoking *CAPBAK/X*

INFORMATION: This chapter shows you how to invoke *CAPBAK/X*.
SEE ALSO: *Installation Instructions*; Chapter 2, “Quick Start”.
LEVEL: This chapter is intended for beginning-to-intermediate users.

4.1 stw Command

To invoke *CAPBAK/X* from the **TestWorks** window:

1. In your current working directory, type `stw`.
2. The **TestWorks** window pops up.
3. Select the **Regression** button.
4. The **STW/Regression** window pops up.
5. Select *CAPBAK/X*.
6. The *CAPBAK/X* window pops up.

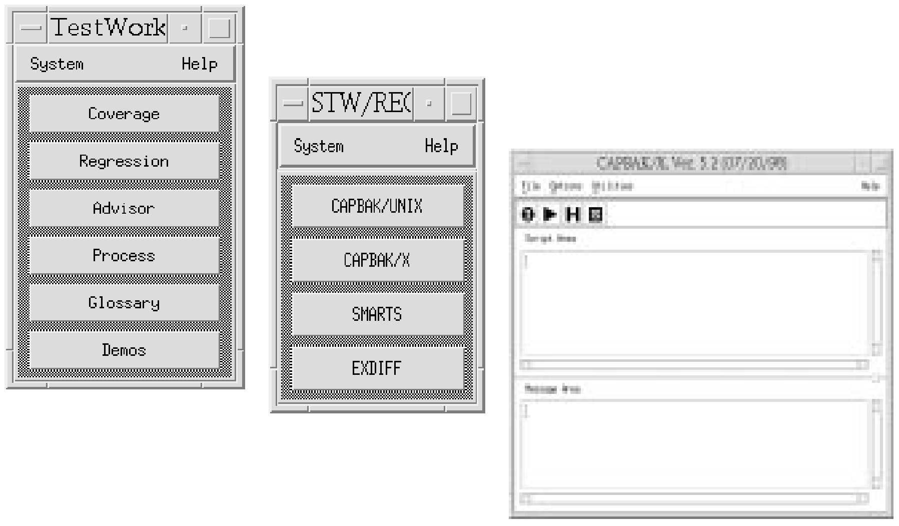


FIGURE 46 TestWorks menus

4.2 Environment Variables

Before you invoke *CAPBAK/X*, make sure you are in the X Window System running a window manager (**mwm**, **olwm**, etc.).

4.2.1 SPARC OpenWindows

If you are running under the SPARC OpenWindows environment, make sure to invoke the X11 / NeWS server with the command:

```
openwin -defeateventsecurity
```

This option is necessary for *CAPBAK/X* operation.

4.3 Xcapbak5 Command

As an alternative to invocation from the **TestWorks** window, you may invoke *CAPBAK/X* from an xterm window in your current working directory with the following command:

Xcapbak5

The *CAPBAK/X* window pops up. It allows you to perform all *CAPBAK/X* operations, including recording / playing back a session, looking at captured images, and getting system information (x / y offsets, window / pointer screen location, etc.).

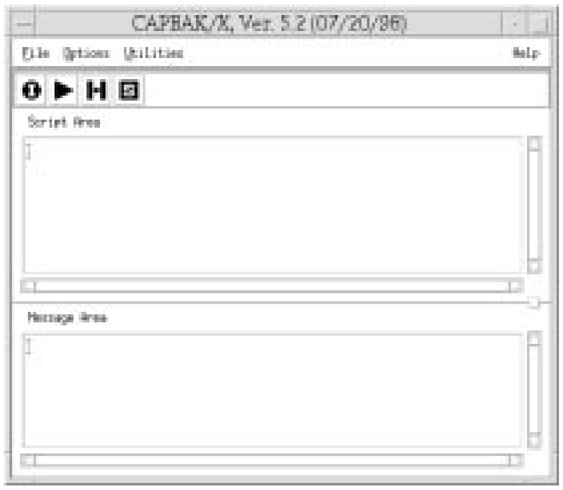


FIGURE 47 *CAPBAK/X* invocation window

4.4 Command-Line Invocation

You can run scripts recorded by *CAPBAK/X* from the command line in a variety of ways.

Prior to playing back a recording, if you are not in the directory from which the test was recorded, you must set the project directory to the that original directory. You can specify the project directory for your current test by using the command:

```
setenv CBX_PROJECT_DIR directoryname
```

prior to beginning command-line playback. Another option for setting your project directory is to use the **`-PD`** switch (see following section).

4.4.1 Xplabak5 command

The **`Xplabak5`** command reads a keysave file and plays back the captured keystrokes, mouse movements, and images.

Like the graphical user interface version of playback, it utilizes easy-to-use function keys. Please note, however, you can only use the function keys listed below for **`Xplabak5`**:

F8	To pause / resume a session (toggles).
F9	To terminate a session.

(The above function keys are the default settings; you can re-configure them permanently by modifying the SR file, or re-configure for the current session only by using the **Set Function Keys** option.)

The session recording begins after you type in the **`Xplabak5`** command and any syntax in the invocation window. A message box will pop up and prompt you:

```
Start a play back session?
```

Click on **OK** and the playback session will begin. Once playback begins, you should not interrupt with the keyboard or mouse, with the exception of the **F8** and **F9** function keys described above.

If you attempt to interrupt in any other way, playback may be unreliable and images may not synchronize.

Playback ends when a message box pops up and prompts you:

```
Play back complete.
```

Below is the required syntax for **`Xplabak5`**:

```
Xplabak5 -k basename.ksv [options]
```

where

-k basename.ksv

Keysave File Switch. Recorded keystrokes, mouse activity, and image captures are played back from the specified keysave file **basename.ksv**. This argument is required unless the keysave file is specified in the configuration file.

Following is a description of the command line options.

-AC *color_list*	Specifies the OCR foreground color by name.
-AF *name*	Specify the OCR font type.
-AM	Move Window on ASCII sync.
-AN *name*	Specify the OCR server name.
-AQ	Quit playing back if the expected ASCII string is not found.
-AS *seconds*	Specify the maximum time to wait for an ASCII string.
-AT *int*	Specify ASCII search area type.
-AV *seconds*	Specify the maximum time interval between ASCII grab.
-D *delay*	Delay Multiplier Switch. Uses **delay** as the playback delay factor. Playback rate can be slowed down or speeded up. A factor of 1 causes no slowdown, or “faithful time playback”; A factor of 0 causes the fastest possible playback, i.e. with zero delays.

The value specified is multiplied by the recorded keystroke timing, and the product will be the delay simulated on playback. The *number* parameter can be either an integer or floating point number, for example 1.5, 0.001, 20, 300, 0.5, etc. Negative values are ignored.

WARNING: Too-rapid playback of scripts may interfere with correct operation of the application under test.

`-ES` *milliseconds*	
	Event Synchronization Specification Switch. Specifies the number of **`milliseconds`** other than the default of 30 that playback will wait for synchronization to occur on selected server events. If synchronization does not occur within the specified time, then that sync event will fail. If you want playback to continue in this case, you must turn the **Quit on Sync Error** option off (default is on).
	Since event synchronization is defaulted to on, you must specify 0 for the number of **`milliseconds`** to turn it off.
`-help`	Help Switch. Forces output to show a summary of available switches. This is also the output produced by an illegal command.
`-IQ`	Image Synchronization Quit Mode Switch. Terminates playback, if image synchronization fails.
`-IM`	Move Window on Image Synchronization Switch. Finds the image of the screen which matches the saved baseline file. When the image is found, moves the image to its proper location (as specified in the keysave file).
`-IS` *seconds*	Image Synchronization Specification Switch. Turns on image synchronization and specifies the time-out period that playback waits for a (1) specified image or window to match a recording's drawn image or (2) finds the image of the screen which matches the saved baseline image.
	The default is 30 seconds. This option applies to the playback of a test session that involves the use of the **Hotkey** window's **Sync Image** button or the **F3** or **F4** function keys to define partial images, pull-down menus or window for synchronization.
`-IV` *seconds*	Image Sync Interval Specification Switch. Defines the interval time for image synchronization to search for the image it is trying to match or to find. The default interval is 3 seconds.
`-NF`	Do Not Save window Frame during playback.

-OC | -OQ	Output Synchronization Continue/Quit Mode Switch. These switches determine if playback will continue, or exit if *CAPBAK/X* fails to synchronize on window pop-ups within the number of seconds specified by the **-OS** switch. If the **-OC** switch is set (this is the default), then playback will continue, even though this may mean that synchronization is lost. If the **-OQ** switch is used, then playback will quit after the specified number of seconds.
-OM	Do Not Move Window on Output Sync Switch. This switch does not move pop-up windows, even if they pop up in a different place during playback than they did during the recording session. You should only use this switch when you are certain pop-up windows occur in the proper locations.
-ON	Do Not Use Window Name On Output Sync Switch. Ignores a window's title name during output synchronization.
-OOPD *(msecs)*	Play Retry switch (ObjectMode recording).
-OORD *(msecs)*	Retry Delay (ObjectMode recording).
-OOOT *(msecs)*	Object Timeout (ObjectMode recording).
-OOWT *(msecs)*	Window Timeout (ObjectMode recording).
-OS *seconds*	Output Synchronization Switch. Specifies the number of **seconds** (other than the default of 30) that playback will wait for synchronization to occur on pop-up windows. Output synchronization is crucial because windows are likely to come up at varying locations on-screen each run. If output synchronization is successful, then a pop-up window will automatically move to its proper location. If no synchronization is achieved, then playback will either continue or quit, depending on your choice of either the **-OC** or **-OQ** switch. Since this switch is defaulted to on, you must specify 0 for the number of *seconds* to turn it off. **Note:** Output synchronization may not work properly for applications that are not ICCCM-compliant.

-OX	Offset X Switch. Specifies the offset in the X direction when a window pops up and is moved to its location, but is still offset by its window frame due to the window manager's own parameters.
-OY	Offset Y Switch. Specifies the offset in the Y direction when a window pops up and is moved to its location but is still offset by its window frame due to the window manager's own parameters.
-OOPD *(msecs)*	Specifies the delay between events in ObjectMode playback.
-OORD *(msecs)*	Specifies the time between object searches in ObjectMode.
-OOOT *(msecs)*	Specifies the maximum "time-out" period that *CAPBAK/X* will search for an object.
-OOWT *(msecs)*	Specifies the maximum "time-out" *CAPBAK/X* will wait for a top window to appear.
-PD *directory*	Specifies a project directory into which you save your project data file, if you don't want to use the current working directory.
-q	Quiet Operation Switch. Start-up copyright messages are suppressed.
-QC	Quick Check Switch. Automatically compares response images (images captured during playback) to their corresponding baseline images (images captured during the initial recording).
-QQ *number*	Quit on Number of Diff Errors Specification Switch. Ends playback if the number of times a response image fails to match a baseline image reaches the number specified. The default is 1. This option be used with the **-QC** switch.
-QR *report_filename*	
	Difference Report Specification Switch. Creates a PASS/FAIL results report of expected image and actual image comparison. This option must be used with the **-QC** switch.
-rb	Regenerate Baseline Images. Saves all playback images to baseline images instead of response images. Do not use the **-QC**, **-QQ** or **-QR** switches with this option.

`-Z`	Compression / Decompression Switch. Turns on the image compression and decompression commands defaulted to `compress -f` (for compression) and `uncompress` (for decompression).
`-ZC` *command*	Compression Command Switch. Allows you to change the compression command which is defaulted to `compress -f`. If you wish to override this default, then you can specify an alternative command by naming it as the `command` option. You must use this switch with the **`-Z switch`**.
`-ZU` *command*	Decompression Command Switch. Allows you to change the decompression command, which is defaulted to `uncompress`. If you wish to override this default, then you can specify an alternative command by naming it as the `command` option. You must use this switch with the **`-Z switch`**.

`-pauseresumekey keysym`

Pause Recording Specification. This function key stops the playback session until either the playback is resumed (same key toggles to resume) or the playback is ended. Default = **F8**.

`-endkey keysym`

End Playback Session Specification. This function key terminates a playback session. Default = **F9.**

`-increaseplay keysym`

Increase playback multiplier specification. This function key increases playback rate by 0.1. Default = **F4.**

`-decreaseplay keysym`

Decrease playback multiplier specification. This function key decreases playback rate by 0.1. Default = **F5.**

CHAPTER 5

Setup for ObjectMode Recording

INFORMATION: Setup needed for ObjectMode (widget) recording with *CAPBAK/X*.
SEE ALSO: Chapter 2, “Quick Start”; Appendix D, “Imakefile Sample”.
LEVEL: Intermediate- to advanced-level users.

5.1 Preparing Your Application for Testing (Static Linking)

CAPBAK/X can be used either with statically- or dynamically-linked applications. SR supplies several versions of its library to allow for ObjectMode testing depending on your application and platform. Little or no modification to the application is needed for dynamic applications. Static applications need to be re-linked to support ObjectMode testing.

Platform	Shared Library Support?	Refer to Section...
SunOS4.1.x	Yes	5.1.1
Solaris 2.x SPARC&x86	Yes	5.1.1
HP-UX 9.x	Yes	5.1.2
IRIX 5.x	Yes	5.1.3
AIX	Yes	N/A
SCO-UNIX	No	N/A
OSF/1	Yes	N/A

TABLE 1 Platform-dependent data: linking SR libraries

5.1.1 SunOS(4.1.x) and Solaris (2.x)

To determine if your program is dynamically- or statically-linked, type:

`ldd` *<applicationname>*

This command lists dynamic dependencies of an executable, if any. For example, if you run this command for "motifbur":

```
% ldd motifbur
```

it returns:

```
-lMrm.2 => /usr/lib/libMrm.so.2.4
-lXm.2 => /usr/lib/libXm.so.2.4
-lXt.4 => /usr/lib/libXt.so.4.10
-lXext.4 => /usr/lib/libXext.so.4.10
-lX11.4 => /usr/lib/libX11.so.4.10
-lc.1 => /usr/lib/libc.so.1.8
-ldl.1 => /usr/lib/libdl.so.1.0
```

If **all** three libraries: `-lXm`, `-lXt` and `-lX11` appear in the list, you have a dynamically-linked executable; if not, then it is statically-linked.

5.1.1.1 Dynamic Linking

This section describes how to set up your environment so that it will automatically invoke the correct `Xt` library needed by **Xcapbak5.**

SR supplies an enhanced version of the `Xt` library in:

```
$SR/lib/capbakx/libSRXt.so.
```

If you have the dynamically-linked executable, no modification is necessary. Follow these steps:

1. Set the *SR* environment variable to your installation directory:

   ```
   % setenv SR <installation directory>
   ```

2. Set up a soft link to the SR-supplied library in *$SR/lib/capbakx*:

   ```
   % ldd <app name>
   ```

   The command should list something similar to the following:

   ```
   -lXt.4 => /usr/lib/libXt.so.4.10
   ```

   Please note that `ldd` points to `libXt.so.4.10`

   Set a soft link from the SR library to that file as shown:

   ```
   % (cd $SR/lib/capbakx; ln -s libXt.so libXt.so.4.10)
   ```

Note: The above step needs to be executed only once.

3. Set the *LD_LIBRARY_PATH* environment variable to allow the AUT to search for the *CAPBAK* library during execution-time loading.

   ```
   setenv LD_LIBRARY_PATH $SR/lib/capbakx:$LD_LIBRARY_PATH
   ```

4. Check to make sure that the AUT is ready for ObjectMode testing:

   ```
           % ldd <application name>
   ```

   `ldd` should now list something similar to the following:

   ```
         -lXt.4 => $SR/lib/capbakx/libXt.so.4.10
   ```

   If that is not the case, please make sure that SR environment variable is set and that `$SR/lib/capbakx/libXt.so.4.10` library exists (it's a soft link to: `$SR/lib/capbakx/libSRXt.so`).

5.1.1.2 Static Linking

(See Section 5.1.1.1 - "Dynamic Linking" on page 83.).

5.1.2 HP-UX 9.x

To determine if your program is dynamically- or statically-linked, type:

```
chatr <applicationname>
```

This command lists dynamic dependencies of an executable, if any. For example, if you run this command for "motifbur":

```
% chatr motifbur
```

it returns:

```
shared executable
shared library dynamic path search:
SHLIB_PATH disabled second
embedded path disabled first Not Defined
shared library list:
static motifbur
dynamic /usr/lib/Motif1.2/libXm.sl
dynamic /usr/lib/X11R5/libXt.sl
dynamic /usr/lib/X11R5/libXext.sl
dynamic /usr/lib/X11R5/libX11.sl
dynamic /lib/libc.sl
shared library binding:
deferred
```

If **all** three libraries: `libXm.sl`, `libXt.sl` and `libX11.sl` appear in the list, you do have a dynamically-linked executable; if not, it is statically-linked.

5.1.2.1 Dynamic Linking

This section describes how to set up your environment so that it will automatically invoke the correct `Xt` library needed by **Xcapbak5.**

SR supplies an enhanced version of the Xt library in:

```
$SR/lib/capbakx/libSRXt.sl.
```

If you have the dynamically-linked executable, you may or may not need to modify your application. Follow these steps:

1. Set the *SR* environment variable to your installation directory:

   ```
   % setenv SR <installation directory>
   ```

2. To determine if you need to modify your dynamically-linked executable for ObjectMode testing, pay attention to the line that starts with *SHLIB_PATH* in the output of `chatr` <*app name*> above. If it says

   ```
   SHLIB_PATH disabled ...
   ```

   you will need to modify your application. Otherwise go to Step 4.

3. You need to turn on or enable the *SHLIB_PATH* option. If this option is enabled in an executable, the executable will look for shared libraries in directories listed in the *SHLIB_PATH* environment variable. To enable this feature, you have the option of modifying the executable using the `chatr` command or re-linking the application with a linker option:

   Change the program’s internal attributes using the chatr command above with the “+s enable” flag on so it will locate the shared libraries in the directory path list specified by the *SHLIB_PATH* environment variable:

   ```
   % chatr +s enable motifbur
   ```

   which will the change the SHLIB_PATH line to “enabled”, giving the following message:

   ```
   SHLIB_PATH enabled...
   ```

   or,

   Add the “+s” option to linker as shown in the following example.

   ```
   % cc -o motifbur motifbur.o -Aa +ESlit -Wl,+s -
   lMrm -lXm -lXt \-lXext -lX11
   ```

   Note that **`-Wl`**, tells the compiler to pass the arguments after the compass to the linker.

4. Check with the `chatr` command:

   ```
   % chatr motifbur
   ```

   It should show that the *SHLIB_PATH* option is enabled:

   ```
   SHLIB_PATH enabled ...
   ```

Note: The above step needs to be executed only once.

Set the *SHLIB_PATH* environment variable to allow the AUT to search for the *capbakx*library during execution-time loading.

If the *SHLIB_PATH* environment variable is not set previously, do:

```
% setenv SHLIB_PATH $SR/lib/capbakx
```

else

```
% setenv SHLIB_PATH $SR/lib/capbakx:$SHLIB_PATH
```

At runtime, the dynamic loader will know to pick up the SR-supplied Xt library.

If that is not the case, please make sure that SR environment variable is set and that `$SR/lib/capbakx/libXt.sl` library exists.

5.1.2.2 Static Linking

If you have a statically-linked executable, SR provides you with a library that, when linked with your application, will allow the application to be tested in ObjectMode.The library is provided in two forms: archive library and object file. They are named `libSRcapbak.a` and `capbak.o`, respectively.

Two Options for Static Linking

Option One

Include only the necessary module and then explicitly load SR library at runtime (by linking in an object file (`capbak.o`)which will explicitly load the shared library at runtime).

Note: This option is available only on systems that support dynamic/ shared library.

For Option One, follow these steps:

1. Set the SR environment variable to your installation directory:
   ```
   % setenv SR <installation directory>
   ```
2. You will need to modify your makefile to include the SR-supplied object file. Add:
   ```
   $(SR)/lib/capbakx/capbak.o
   ```
   to your list of object files. For example (added information in bold),
   ```
   cc -o motibur motifbur.o $(SR)/lib/capbakx/capbak.o\
   -lMrm -lXm-lXt -lX11
   ```

Option Two

Include the complete SR library into your application by linking with the archive library `libSRcapbak.a`.

We recommend that you choose the first option. Doing so will not require you to re-link your application when future SR libraries become available.

Follow these steps for Option Two:

1. Set the SR environment variable to your installation directory:
   ```
   % setenv SR <installation directory>
   ```

2. Modify your makefile to include SR library before you include the Xt library. Add:

   ```
   $(SR)/lib/capbakx/libSRcapbak.a
   ```

   to your link command right before the Xt library, for example (added information shown in bold):

   ```
   cc -o motifbur motifbur.o -lMrm -lXm\
    $SR)/lib/capbakx/libSRcapbak.a -lXt -lX11
   ```

For a more detailed example of how static and dynamic linking are done, please refer to the example Imakefile for **motifburger** application in Appendix D *"Imakefile Sample from motifburger"*.

5.1.3 Irix 5.x

To determine if your program is dynamically- or statically-linked, type:

```
elfdump -Dl <applicationname>
```

This command lists dynamic dependencies of an executable, if any. For example, if you run this command for "motifbur":

```
% elfdump -Dl motifbur
```

it returns:

```
***LIBRARY LIST SECTION***
Name        Time-Stamp          CheckSum
Flags Ver.
.liblist
libMrm.so.1  Aug 22 17:45:59 1993
0x50f93f2b  NONE Based on OSF/Motif
1.2.2#sgi1.0
libXm.so.1   Aug 21 18:32:53 1993
0x2f014e65  NONE Based on OSF/Motif
1.2.2#sgi1.0
libXt.so     Aug 18 21:31:00 1993
0xb69b1b36  NONE sgi1.0
libXext.so   Aug 18 21:32:21 1993
0x0e5d6fb5  NONE sgi1.0
libX11.so.1  Aug 24 20:44:52 1993
0x1054938e  NONE sgi1.0
libgen.so    Aug 18 21:27:02 1993
0x33adab00  NONE sgi1.0
libc.so.1    Aug 18 21:19:38 1993
0xdf9c0716  NONE sgi1.0
```

If **all** three libraries: `libXm.so`, `libXt.so`, and `-libX11.so` appear in the list, you do have a dynamically-linked executable; if not, then it is statically-linked.

5.1.3.1 Dynamic Linking

This section describes how to set up your environment so that it will automatically invoke the correct `Xt` library needed by **Xcapbak5.**

SR supplies an enhanced version of the Xt library in:

```
$SR/lib/capbakx/libSRXt.so.
```

If you have the dynamically-linked executable, no modification is necessary. Follow these steps:

1. Set the *SR* environment variable to your installation directory:

```
% setenv SR <installation directory>
```

Note: The above step needs to be executed only once.

---

2. Set the *LD_LIBRARY_PATH* environment variable to allow the AUT to search for the *capbakx* library during execution-time loading.

```
setenv LD_LIBRARY_PATH_\
 $SR/lib/capbakx:$LD_LIBRARY_PATH
```

5.1.3.2 Static Linking

Refer to Section 5.1.2.2. on page 87

5.2 Verification Messages for Proper Linking

Whether you choose the static or dynamic linking, your application will indicate if it finds the required library for **Xcapbak5** operation. You will get a message similar to the following:

```
Application Under Test ‘motifbur’ is linked for
widget-level testing.
```

In addition, if you’re using the static linking approach, you will get a message:

```
Using shared lib: $SR/lib/capbakx/libSRcapbak.so
```

If you do not get such messages, your application cannot locate the required library. Please make sure such a library exists and you have the SR environment variable set properly.

5.3ObjectMode Limitations

ObjectMode capture has certain limitations, among them:

1. It works only on Xt-Intrinsics-based toolkits (i.e. Motif, Athena widget sets).
2. It currently works on only one application at a time.
3. It does not currently handle drawing or painting applications (for these applications, you must use the true-time mode of recording).
4. It currently only supports Motif 1.1 and 1.2 widget sets.
5. It does not support the use of accelerators (“short-cut” key combinations).
6. It does not support double-clicking (you must “click and drag” to choose options, etc.). This is true even in the case of “pop-up” and “pull-down” menus, where there is a longer time interval between clicks than usual.
7. Limited support for keyboard traversals (cursor-up, cursor-down arrow keys, for example).
8. If you are working on an HP machine, it may or may not recognize the F9 function key as the “End” key during record / playback. If this is the case, change the default setting in the SR file to F10.
9. At present, on the SGI platform, *CAPBAK/X* cannot capture images generated using the system’s graphic libraries.
10. If *CAPBAK/X* is not picking up the proper resource file, you may have to set it using the **`xrdb`** command:

    To set the application resources properly, type in the command line,

    (for OpenWindows):

    ```
    xrdb -merge $openwinhome/lib/app-defaults/SR
    ```

    (for X11):

    ```
    xrdb -merge /usr/lib/X11/app-defaults/SR
    ```

    This will establish the proper settings to do ObjectMode testing. After setting these defaults, you can invoke *CAPBAK/X*. Please see *Installation Instructions* for further information on proper setups for running *CAPBAK/X* and other TestWorks tools.

CHAPTER 6

Playback Synchronization Issues

INFORMATION: Cause and resolution of playback synchronization issues.
SEE ALSO: Section 1.2.1, “Synchronization Aspects”; Appendix A, “Customizing *CAPBAK/X*”.
LEVEL: This chapter is intended for intermediate-to-advanced users.

6.1 Problem: Loss of Synchronization

In the X Window System environment, an application’s windows are likely to come up at varying locations, and speed may vary between capture and playback sessions.

For this and other reasons, the playback mechanism that is in use can operate in such a way that the coherence between the recording and the response is lost. This loss of synchronization results in either chaotic behavior of the application, possible loss of essential test output information, or both.

Regardless of cause, loss of synchronization is serious: you risk losing valuable test time as well as coherent test processing.

How de-synchronization can happen — and what you can do about it — are important factors in implementing your automated test regression systems.

6.1.1 Applications and Synchronization

The most common kind of de-synchronization problem arises because the application itself or the environment is not behaving reproducibly. Two of the most common causes for this are described next.

6.1.1.1 Window Managers

In some X Window Systems, the window manager actually introduces non-reproducibility on purpose; it is often labeled as a feature. To achieve reproducibility, this feature must be disabled.

For example: The OSF/Motif window manager (**mwm**) has a feature that avoids placing a new window exactly over an existing window. The idea is that the user will know if, by accident, there are two new windows created and not just one. If this feature did not exist a user might inadvertently create two overlapping windows.

To illustrate this, try the command **`xterm &; xterm &`**. You should get two new windows which do not overlap exactly.

This feature, called the *auto client place* feature, is done with an internal algorithm that randomizes, in part, the selection of a new window's screen location. When this feature is active, one can virtually predict loss of synchronization by *CAPBAK/X* during playback!

This can be resolved in one of two ways:

- Disable the **mwm** auto client place feature by adding the following line to your `.Xdefaults` file:

  ```
  Mwm*clientAutoPlace: False
  ```
- Use the *CAPBAK/X* automatic output synchronization feature to control pop-up windows.

To handle this problem, *CAPBAK/X* will wait a number of seconds (defaulted to 30) for synchronization to occur. If it does, the window will automatically move to the same location it was in during the recording session. If it fails to synchronize, playback will either exit or continue depending on what is specified.

To activate output synchronization from the *CAPBAK/X* window:

1. Click on the **Options** menu.
2. Select the **Set Output Sync Options** submenu.

   The **Output Sync Options** window pops up.
3. Make sure the output synchronization switches you need are turned on. Please refer to Section 3.4.1 for a list of the options.
4. If you want playback to exit if synchronization fails, leave the **Quit on Output Sync Error** option on.

   Anytime a window manager's window pops up during a recording session, *CAPBAK/X* automatically generates a

   ```
   cb_sync_win_activate  ["window_name"X Y width
      height ]
   ```

   statement in the keysave file.

If you invoked an xterm window during a test session, you may get a statement similar to the one below:

```
cb_sync_win_activate ["xterm"81 54 484 316 ]
```

When the test is played back, this function call is interpreted, and *CAPBAK/X*:

- Waits for a window that has an `xterm` banner, a width of 484 pixels and a height of 316 pixels to appear.
- Checks that the window is brought up in the same location as during the recording session (with xy coordinates 81, 54).
- If the **Displace Window On Output Sync** option is on, repositions the window if the coordinates are different.

If you are running playback from the command line, turn on the **-OS** switch and either the **-OC** switch to continue if synchronization fails or the **-OQ** switch to quit if synchronization fails. Please refer to Section 4.4 for further information.

6.1.1.2 Applications

When the application itself does things at random the problem is more serious.

An example would be when your application opens a pop-up window and chooses — for variety — to position it at location that is computed to never be the same spot. This is similar to the problem encountered with the window manager. To handle this problem, you can do one of three things:

- Fix an application's pop-up location by changing the application's resource file. This is normally located in

  `/usr/lib/X11/app-defaults/app_class_name`

  or

  `$OPENWINHOME/lib/app-defaults/app_class_name`

  for SPARC OpenWindows. This may not be true for all applications.
- Use *CAPBAK/X*'s automatic output synchronization options as described in Section 3.4.1.
- When a window pops up during a recording session, click on its frame to bring it into focus.

In addition to application pop-up synchronization problems, applications do not always respond at the same speed during the capture and playback phases. In other words the time that elapsed until a window is redrawn may vary from one test run to another. To handle this:

- Use *CAPBAK/X* image synchronization features to hold back execution until a window or partial image is redrawn. This approach assumes that the user records a set of images — during the recording session — that will act as the synchronization method during playback.

  CAPBAK/X pauses test execution for a number of seconds (defaulted to 30 seconds) to wait for the exact image to occur at the exact place on the screen, or to find the image of which matches that of the save baseline image.

If the image is not found, playback will either exit or continue, depending on what is specified.

To activate image synchronization from the *CAPBAK/X* window:

1. Click on the **Options** menu.
2. Select the **Set Image Sync Options** submenu.
3. The **Image Sync Options** window pops up.
4. Make sure the image synchronization switches you need are turned on. Please refer to Section 3.4.1 for a list of the options.
5. If you want playback to exit if image synchronization fails, leave the **Quit on Image Sync Error** option on.
6. If you want playback to wait for an image to match a baseline's drawn image, do *not* turn on the **Displace Window on Image Sync** option. Turn the **Displace Window on Image Sync** option only if you want playback to find the image which matches that of the saved baseline image and moves it to the right location. In other words, playback finds an image but does not try to match it exactly.
7. During a recording session, you can specify an image or window to match or to be found for image synchronization during playback by using the **Hotkey** window's **Sync Image** button or the **F3** function key for partial images and pull-downs or the **F4** function key for windows.

   Please see Section 5.4.1 for complete instructions on specifying images or windows for synchronization.
8. A `cb_sync_area_image` *(X Y width height "basename.xxx"*) statement is generated in the keysave file, where *basename* represents the name of the test, and *xxx* shows the original sequence in which the image for synchronization was captured.

   Playback interprets this statement and tries to locate the image and match it.

   - Adjust timing during playback with the **Delay Multiplier** option in the **Other Options** window if you are using the GUI. You should experiment with different playback speeds and stay with a comfortable, constant speed. If you use the **F4** and **F5** function keys to change the speed *during* playback, you may jeopardize synchronization.

You should try to find a speed that is not too fast to de-synchronize a session or too slow to try the patience of the tester. Typically, a delay multiplier of < 0.5 will overdrive the application and a value of > 20.0 will probably assure synchronization, but will be too slow for even the most patient tester.

6.2 Problem: Offsets Incorrect

The X and Y offsets required by *CAPBAK/X* can vary from platform to platform and window manager to window manager. The following section shows how to determine the correct offsets for your system.

6.2.1 Automatic Determination of Machine-Dependent Offsets

The SR's **CB.offset** utility computes the appropriate border offsets for your particular windowing system and returns this information in a form that is appropriate to be added to the *$SR* file (see below). In most cases **CB.offset** will work correctly on your windowing system as it is supplied, i.e. as an executable.

To determine the offsets from the **Xcapbak5** GUI,

A. Click on **Utilities** to pull-down the menu.

B. Click on **Miscellaneous**.

C. Select **Get Offset Resource** from the menu.

Four lines of code will be returned as follows:

```
capbakX5*option*outputSyncXoffset: 5
capbakX5*option*outputSyncXoffset.value: 5
capbakX5*option*outputSyncYoffset: 25
capbakX5*option*outputSyncYoffset.value: 25
```

Section 6.2.3 lists some values of the x and y offsets for typical windowing systems on various platforms.

After you have determined the proper x,y offset values for your environment, you must add this information to your SR file in order for *CAPBAK/X* to function properly. Simply locate the lines above in your SR file and edit them accordingly.

The resulting coordinates can then be set into the *SR* file and another

`xrdb -merge <`*location*`>/SR`

command issued.

Try another recording / playback session for verification of correct offsets.

6.2.2 Common Settings

Shown in the table below are typical x,y offset values for various platforms and window managers.

Platform	Window Manager	x, y offsets
HP-9000	**OLWM**	6, 35
	HPVUE	8, 24
	MWM	8, 25
Solaris 2.1	**MWM**	11, 28
Solaris 2.3	**MWM**	8, 24
SUN/OS	**MWM**	0, 0
	OLWM	1, 1
RS 6000	**MWM**	0, 0
SGI Indigo		8, 29
SGI Iris		0, 0
NCR 3300/UNIX SVR 4		7, 24

TABLE 2 x,y offsets for Common Platforms

CHAPTER 7

Optical Character Recognition (OCR)

INFORMATION: OCR extends the life of keysave files by tolerating minor application changes, extracting characters and finding the location of a character string.

SEE ALSO: *STW Installation Instructions*, Chapter 3.
LEVEL: Intermediate to Advanced users.

7.1 Platform Availability

The character recognition capabilities work on the following platforms:

- SPARC running Sun/OS with X11 or OW
- SPARC running Solaris
- RS/6000 under AIX 3.2.*n*
- HP-9000/7xx under HP-UX 9.*n*

7.2 Understanding the OCR Technology

Software Research has licensed OCR (Optical Character Recognition) technology from Xerox Imaging Systems (XIS) to provide generalized character recognition capabilities. The *CAPBAK/X* implementation of this technology lets you:

- Capture a character string for playback synchronization.
- Capture the characters from an area of the screen or an entire window to a file.
- Extract characters from a saved image file (`*filename*.xwd`) or from the current screen.
- Locate a specified character string in saved image file (`*filename*.xwd`) or from the current screen.

Note: Prior to using the OCR technology, you *must* install the OCR server. Please see the *Installation Instructions* (Chapter 3) for further information.

7.3 OCR Restrictions

When you are using character recognition, please be advised of the following restrictions.

- OCR can extract up to 10,000 characters.
- Character synchronization capturing is limited to one line, five words.
- Software Research, Inc. has trained the OCR software to recognize 450 of the most common fonts in X.
- Characters must be at least 10 point size. If you are on the HP, characters must be at least 12 point size.

7.4 Character-Based Synchronization

Output generated by the application-under-test (AUT) may vary at any given time. Application changes, such as button location, fonts and background colors and open-file menu list output, cause differences between recording and playback sessions that the user may want reported.

In order for these kinds of changes to be tolerated *CAPBAK/X* offers character synchronization. Character synchronization is accomplished with the keysave file `cb_sync_area_ascii` function.

This function reads a line of text from an area of the screen or from a window, and returns that text as a character string that is to be located during playback.

A character string is captured and *CAPBAK/X* searches for it in one of the following locations:

1. The topmost window.
2. The window that the mouse is in.
3. The entire screen.

Because the OCR technology has to search the specified area for characters during playback, searches on the entire screen are slow. The location of the search can be set with one of the **Displace Pointer On ASCII Sync** sub-options in the GUI's **ASCII Sync Options** window or the command line's **`Xplabak5 -AT`** ***`int`*** parameter. Please refer to other sections for further information (See Section 7.4.2 - "Selecting Character Synchronization Options From the GUI" on page 107.), (See Section 7.4.3 - "Selecting Character Synchronization Options From the Command Line" on page 111.).

While all three kinds of character synchronization searches handle minor application changes, changes in window size are *not* tolerated. For instance, if the font size were substantially changed, the size of a window might also change accordingly. *CAPBAK/X* automatic synchronization on pop-up windows and menus would fail because it is reliant on the x, y coordinates and on the width and height remaining the same.

7.4.1 Synchronizing On a Character String

During your recording session you can synchronize on a character string by using the **`Ctrl`** and **`F3`** keys.

You should synchronize on *each* location you think will change in an application. If you plan on interchanging the locations of an **OK** and a **Cancel** button in a file selection window, for instance, you should capture both the **OK** and **Cancel** character strings for playback synchronization to occur.

Note: If you are using *CAPBAK/X* from the graphical user interface (GUI), you do not have to set any special option for character synchronization to work.

To synchronize on a character string:

1. When you want to synchronize on a character string, **press the `Ctrl` and `F3` keys at the same time and then lift them up**. The recording session pauses and the screen pointer becomes a cross-hairs.
2. To bound the characters to be read, move the cross-hairs to one corner of the character string to be captured.
3. Press and hold down the mouse button, drag the mouse until the rectangle traced contains the entire string to be included. A string should consist of whole words, rather than segments of a word. A word is any text that has a space before and after it.
   - It is recommended that you bind the rectangle as closely as possible to the string.
   - Make sure to capture whole characters; the OCR technology cannot interpret partial characters.
   - Character strings must be in one line and no more than five words.
   - A character string should have blank space on both sides; in other words, you cannot capture “part” of a bigger word.
4. When the entire character string is in the rectangle, release the mouse button.

   The status window states:

   ```
       Sync ASCII partial saving...
   ```

The **OCR Verify** window pops up with the character string you selected shown in the display area.

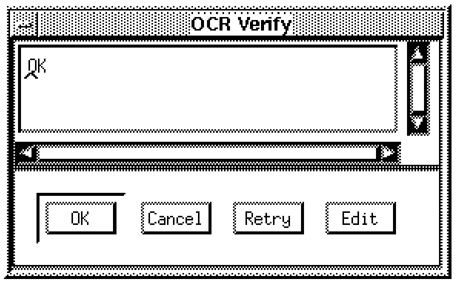


FIGURE 48 OCR Verify Window

- Use the scroll bars to maneuver throughout the **OCR Verify** window.
- Click on **OK** if the character string displayed is what you want to synchronize on.
- Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
- Click on **Retry** if the character string is not the string you wanted. The status window states:

  `Sync ASCII partial...retry`.

  The screen pointer turns into a cross-hairs so another character string can be captured. Please follow steps 2-4 to capture another character string.
- Click on the **Edit** button to modify the selected character string. When this button is selected, it toggles to **Done** and the other buttons are dimmed. Move the mouse pointer to the location where you want to edit and click on the mouse button. A cursor will appear, indicating you can begin editing. Terminate an editing session by clicking on **Done**. This action will toggle the button back to **Edit** and the other options will become available.

5. If the **OK** button was selected, *CAPBAK/X* signals capture of the character string with a bell and the status window then reads:

```
Sync ASCII partial saving...complete.
```

CAPBAK/X generates a `cb_sync_area_ascii` *`("string" X Y width height`*`)` event statement in the keysave file, indicating the string to be synchronized on when the test is played back.

Here, *X*, *Y* are the coordinates of the corner at which you started drawing the rectangle for the string; *`width`* and *`height`* are the dimensions of that rectangle; and*`, string`* is the captured character string.

The mouse pointer will then automatically move to the center of the string on which you synchronized.

Note: Until the next sequence of “button down” and “button up” is recorded (i.e., `cb_key_down` and `{cb_key_up` in the keysave file, *relative motion* is recorded (i.e., `cb_mouse_move_abs (`*X Y* *`delay`*) into the keysave file. Motion is relative to where mouse was before the movement. When the mouse pointer is first moved, its motion is relative to the center of the string. Each subsequent movement is then relative the last mouse movement.

Relative motion is key to OCR synchronization. The mouse movement during playback will be based on — that is, synchronized on — the new location of the ASCII string you synchronized on during the recording session.

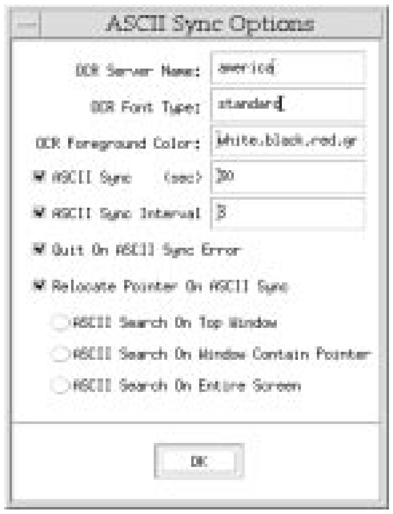
6. Continue your recording session as you normally would.

7.4.2 Selecting Character Synchronization Options From the GUI

Prior to playback you may want to select some of the available character synchronization options with the **ASCII Sync Options** window.

To initialize the **ASCII Sync Options** window:

1. Click on the **Options** menu.
2. Select the **Set ASCII Sync Options** sub-menu.
3. The **ASCII Sync Options** window pops up.



ASCII Sync Options

OCR Server Name: america
OCR Font Type: standard
OCR Foreground Color: white,black,red,gr
[✓] ASCII Sync (sec) 30
[✓] ASCII Sync Interval 3
[✓] Quit On ASCII Sync Error
[✓] Relocate Pointer On ASCII Sync
○ ASCII Search On Top Window
○ ASCII Search On Window Contain Pointer
○ ASCII Search On Entire Screen

FIGURE 49 Character Sync Options Window

You can select any of the following options:

- **OCR Server Name:** Specifies the name of the host machine where the OCR license server is installed. During the installation procedure, the host machine's default is automatically appended to the *SR* file. Please refer to Chapter 4 of the *Installation Instructions* for further information on appending the default to the *SR* file.
- **OCR Font Type:** Specifies the name of the font family for character synchronization. During the installation procedure, the default font family is automatically placed in the *SR* file. Please refer to Chapter 4 of the *Installation Instructions* for further information on appending the default to the *SR* file.
- **OCR Font Colors:** If the OCR software cannot find the colors in the SR file, then it looks for text in the specified colors; up to four colors can be listed. The colors must be legal X-color names (see the *X User's Guide* for a description of legal color names). If there

is more than one, then the color names must be separated by a comma, with no space in between them.

- **ASCII Sync:** Defines the time-out period that playback waits for a specified character string to be found. The default is 30 seconds. Unless the **Quit On ASCII Sync Error** option is turned on, playback proceeds even if the character string is not found.
- **ASCII Sync Interval:** Defines the interval time for playback to search for the character string it is trying to find. The default interval is three seconds.
- **Quit On ASCII Sync:** Terminates the playback session, if character synchronization fails.
- **Displace Pointer On ASCII Sync:** The option performs two functions:

  If a character string moved and is found, this option moves the mouse pointer to the new location.

  Changes where the search for the character string is done in respect to the screen.

Note: If this option is not selected, playback restricts the area of the character string search to the area of the rectangle captured. If this option is turned on and none of the below sub-options are selected, the location of the search is defaulted to the top window, or “focus” window.

ASCII Search On Top Window

Searches for a captured character string in the screen’s topmost window, sometimes referred to as the “focus” window.

ASCII Search On Window Containing Pointer

Searches for a captured character string in the window that contains the mouse pointer.

ASCII Search On Entire Screen

Searches for a captured character string throughout the entire screen. *This may take the system a long time.*

You can change the defaults for any of the above options by editing the following lines in the *SR* file, which is located in */usr/lib/X11/app-defaults* or *$OPENWINHOME/lib/app-default*s for Sun SPARC running Open Windows, Version 3.

```
capbakX5*ocrServerName: host_name
capbakX5*option*ocrServerName.value: host_name
capbakX5*ocrFontType: standard
capbakX5*option*ocrFontType.value: standard
capbakX5*ocrForegroundColor:
white,black,red,green
capbakX5*option*ocrForegroundColor.value:
white,black,red,green
capbakX5*option*asciiSyncSwitch.set:True
capbakX5*option*asciiSyncDelay.value: 30
capbakX5*option*asciiSyncIntervalSwitch.set:
True
capbakX5*option*asciiSyncIntervalDelay.value: 3
capbakX5*option*asciiSyncQuit.set: True
capbakX5*option*asciiSyncMove.set: False
```

FIGURE 50 Character Synchronization SR File Defaults

`capbakX5*ocrServerName scotland`

`capbakX5*option*ocrServerName.value: host_name`

The **OCR Server Name** is defaulted to the specified `host_name`.These lines are appended to the *SR* file during installation. Please see Section 3.2.1 of the *Installation Instructions* for information on appending these lines.

`capbakX5*ocrFontType standard`

`capbakX5*option*ocrFontType.value:font_type`

The **OCR Font Type** is defaulted to the `font_type`. These lines are appended to the *SR* file during installation. Please see Chapter 4 of the *Installation Instructions* for information on appending these lines.

`capbakX5*ocrForegroundColor:white,black,red,green`

```
capbakX5*option*ocrForegroundColor.value:white,black,
                red,green
```

The **OCR Font Colors** are defaulted to `white, black, red, green`. Change to the names of the colors of the type you want to recognize (maximum four colors).

`capbakX5*option*asciiSyncSwitch.set:True`

The **ASCII Sync** option is defaulted to on. Change `True` to `False` to default this option to off.

```
capbakX5*option*asciiSyncDelay.value: 30
```

The **ASCII Sync** time-out period is defaulted to 30 seconds. Change `30` to the number of seconds you want for the default.

```
capbakX5*option*asciiSyncIntervalSwitch.set:True
```

The **ASCII Sync Interval** option is defaulted to on. Change `True` to `False` to default this option to off.

```
capbakX5*option*asciiSyncIntervalDelay.value: 3
```

The **ASCII Sync Interval** search time is defaulted to 3 seconds. Change 3 to the number of seconds you want for the default.

```
capbakX5*option*asciiSyncQuit.set:True
```

The **Quit On ASCII Sync** option is defaulted to on. Change `True` to `False` to default this option to off.

```
capbakX5*option*asciiSyncMove.set:True
```

The **Displace Pointer On ASCII Sync** option is defaulted to on. Change `True` to `False` to default this option to off.

7.4.3 Selecting Character Synchronization Options From the Command Line

Character synchronization options can also be set from the command line. The options below have been added to the **Xplabak5** options.

7.4.3.1 Xplabak5 Character Synchronization Options

Following is a description of the character synchronization command line options.

-AC *color,color*

OCR Font Colors Specification Switch. Specifies the name of the colors that the OCR technology looks for if it cannot find black or white type in the specified area. The colors must be legal X-color names (see the *X User’s Guide* for a description of legal color names.) If there are two colors, then the color names must be separated by a command, with no space in between them.

-AF *name*

OCR Font Type Specification Switch. Specifies the name of the font family for character synchronization. The default *name* is automatically appended to the *SR* file during the installation procedure. If *name* is not appended to the *SR* file, then the default is the default font family is *standard*, which contains 60 of the most common X fonts. *name* should be the same font family you set with this option with the **Xplabak5 -AF** command.

-AM

Move Pointer on ASCII Synchronization Switch. If a character string is moved and found, this option moves the mouse pointer to the new location of the character string.

-AN *name*

OCR Server Specification Switch. *name* specifies the name of the host machine where the OCR license server is installed. The default *name* is automatically appended to the *SR* file during the installation procedure. If name is not appended to the *SR* file, then the default is the current name of the host machine you are on.

-AQ

Quit on ASCII Sync Error Switch. Terminates the session if playback is unable to synchronize on the character string.

-AS *seconds*	ASCII Sync Time-Out Specification Switch. *seconds* specifies the number of seconds that playback will wait for a specified character string to be found. The default is 30 seconds.
-AT *int*	ASCII Sync Search Area Specification Switch. *int* specifies the region a captured character string is searched. *int* can be one of the following: • (Top=1) Searches for a captured character string in the screen's topmost window, sometimes referred to as the "focus" window. • (Mouse=2) Searches for a captured character string in the window that contains the mouse pointer. • (Root=3) Searches for a captured character string throughout the entire screen. This may take the system a long time • (NONE_WIN=0). If this switch is not specified, playback restricts the area of the character string search to the exact location captured.
-AV *delay*	ASCII Sync Interval Specification Switch. *delay* specifies the interval time (in seconds) for playback to search for the string it is trying to find. The default is three seconds.

7.4.4 Playing Back with Character Synchronization

Prior to playback you should have set any necessary options with the **ASCII Sync Options** window, as described in Section 7.4.2.

When the `cb_sync_area_ascii [`*“string” X Y width height* `]` is encountered during playback:

1. Playback pauses.
2. The cursor moves to the *X*, *Y* coordinate location specified in the keysave file and waits for the string to be searched.
3. The area of the search depends on what you selected in the GUI’s **ASCII Sync Options** window or the command line’s **Xplabak5 -AT** *int* option. Please see Section 7.4.3.1.
4. If the string is found within the bound area you specified, the *CAPBAK/X* window returns with the message

   ```
   ASCII sync...complete.
   ```

   and the cursor moves to the middle of the string found.

7.5 Capturing ASCII Characters

Previously, users could only capture bitmap images. While bitmap images are useful for general playback verification, they can be inaccurate in certain cases where the output of an image remains the same but the font has changed. In other words, bitmap captures are concerned with pixel differences, not the actual values within the captured image. *CAPBAK/X* ASCII character captures, on the other hand, record only the values from an area of the screen or a window. During playback the values are captured again and compared for differences with the recording's saved values.

Sections 7.5.1 through 7.5.6 illustrate the different techniques to capture ASCII characters from an area of the screen or from entire windows.

7.5.1 Screen Area Character Capturing — Using the Save ASCII Partial Button

To capture the values from an area of the screen using the **Hotkey** window's **Save ASCII Partial** button:

1. When you want to save the values from a particular area of the screen during a recording session, press the **F1** function key.

   The recording session pauses and the **Hotkey** window pops up.

   The status window states:

   ```
   Command Mode. Select option from menu.
   ```
2. Select the **Save ASCII Partial** button.

   The screen pointer becomes a cross-hairs.
3. To bound the characters to be read, move the cross-hairs to area containing the text and move the cross-hairs to one corner of the character string.
4. Press and hold down the mouse button, drag the mouse until the rectangle traced contains the entire area of values to be included. Make sure you captured whole characters. Otherwise, the OCR will be unable to interpret these characters.
5. When the character string with whole character values is in the rectangle, release the mouse button.

   The status window states:

   ```
   ASCII partial saving...
   ```

The **OCR Verify** window pops up with the selected character string shown in its display area.

- Use the scroll bars to maneuver throughout the **OCR Verify** window.
- Click on **OK** if the character string displayed is what you want.
- Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
- Click on **Retry** if the screen area's values are not what you intended to capture. The status window states: `ASCII partial saving...retry`. The screen pointer turns into a cross-hairs so you can capture another screen area's values. Please follow steps 3-4 to capture another character string.
- If necessary, click on the **Edit** button to modify the screen area's captured values. When this button is selected, it toggles to **Done** and the other buttons are dimmed. Move the mouse pointer to the location where you want to edit and click on the mouse button. A cursor will appear, indicating you can begin editing. Terminate an editing session by clicking on **Done**. This action will toggle the button back to **Edit** and the other options will become available.

6. If the **OK** button was selected, *CAPBAK/X* signals capture of the characters with a bell and the status window states:

```
ASCII partial saving...complete.
```

The characters are saved as *basename.xxx*, saved to a `BSL_a` directory to indicate it is an ASCII character baseline file; *xxx* shows the original sequence in which the file was captured. You can view this file with any text editor.

CAPBAK/X generates an `{cb_save_ascii ["test name", sequence,` *X Y* *`width height`*`]` event statement in the keysave file. Here, *X, Y* are the coordinates of the corner at which you started drawing the rectangle and *`width`* and *`height`* are the dimensions of that rectangle.

The OCR technology also creates a temporary working file of the relative motion named `ocr.out`.

7. Use the **Hotkey** window as you normally would to resume recording or to issue other commands.

   During playback the `{cb_save_ascii ["test name", sequence,} [`*`0 X Y width height`*`]` statement is interpreted and a response file of the values is captured named *basename.xxx*, saved in a `RSP_a` directory indicating it is an ASCII character response file; *xxx* shows the original sequence in which the file was captured.

   Playback verifies that *`basename.xxx`* was captured with the following message in the status area of the *CAPBAK/X* window:

   ```
   ASCII partial saving....complete.
   ```

If the *Quick Check* mode of playback is used with the GUI's **Quick Check Options** window or the command line's **Xplabak5 -QQ** option, *`basename`*`.xxx` baseline and response files are automatically compared.

If the character strings compare with no difference, the status window displays:

```
Comparison PASSES on ASCII number #.
Test continues.
```

If the character strings' difference and the GUI's **Quit On Number Of Diff Errors** or the command line's **Xplabak5 -QQ** options are *not* specified, the status window displays:

```
Comparison FAILS on ASCII number #.
Test continues.
```

If the character strings' difference and the GUI's **Quit On Number of Diff Errors** or the command line's **Xplabak5 -QQ** options are specified and the number of errors is equal to or more than the number specified, the status window displays:

```
Comparison FAILS on ASCII number #
Test stopped before completion.
```

Please see Section 3.4.1.4 of the *CAPBAK/X User's Guide* for complete information on the *Quick Check* options available.

7.5.2 Screen Area Character Capturing — Using the Ctrl + F5 Keys

To capture the values from an area of the screen with the **Ctrl** and **F5** keys:

1. When you want to synchronize on a character string, press the **Ctrl** and **F5** keys at the same time and then lift them up.

   The recording session pauses and the screen pointer becomes a cross-hairs.

2. To bound the characters to be read, move the cross-hairs to the area containing the text and move the cross-hairs to one corner of the character string.
3. Press and hold down the mouse button, drag the mouse until the rectangle traced contains the entire string to be included.
4. When the character string with whole character values is in the rectangle, release the mouse button.

   The status window states:

   ```
   ASCII partial saving...
   ```

   The **OCR Verify** window pops up with the character values from the area of the screen you captured displayed.

   - Use the scroll bars to maneuver throughout the **OCR Verify** window.
   - Click on **OK** if the character string displayed is what you want.
   - Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
   - Click on **Retry** if the screen area's values are not what you intended to capture. The status window states: `ASCII partial saving...retry`. The screen pointer turns into a cross-hairs so you can capture another screen area's values. Please follow steps 2-4 to capture another character string.
   - If necessary, click on the **Edit** button to modify the screen area's captured values. When this button is selected, it toggles to **Done** and the other buttons are dimmed. Move the mouse pointer to the location where you want to edit and click on the mouse button. A cursor will appear, indicating you can begin editing. Terminate an editing session by clicking on **Done**. This action will toggle the button back to **Edit** and the other options will become available.

5. If the **OK** button was selected, *CAPBAK/X* signals capture of the character string with a bell and the status window states:

   ```
   ASCII partial saving...complete.
   ```

The characters are saved as *basename.axx*, where *a* indicates that it is an ASCII character baseline file and *xx* shows the original sequence in which the file was captured. You can view this file with any text editor.

CAPBAK/X generates a `{cb_save_ascii ["test name", sequence,` *X Y* *`width height`*`]` event statement in the keysave file. Here, *X*, *Y* are the coordinates of the corner at which you started drawing the rectangle and *`width`* and *`height`* are the dimensions of that rectangle.

The OCR technology also creates a temporary working file of the characters saved named `ocr.out`.

6. Use the **Hotkey** window as you normally would to resume recording or to issue other commands.
7. Continue the recording session.

   During playback the `{cb_save_ascii ["test name", sequence,` *X Y* *`width height`*`]` statement is interpreted and a response file of the characters is captured named *`basename`*`.xxx`, saved to a `RSP_a` directory; `xxx` shows the original sequence in which the image was captured.

Playback verifies that *basename.xxx* was captured with the following message in the status window of the **Record/Play** window:

```
ASCII partial saving....complete.
```

If the *Quick Check* mode of playback is used with the GUI's **Quick Check Options** window or the command line's **Xplabak5 -QQ** option, *basename.xxx* image and response strings are automatically compared.

If the character strings are equivalent, the status window displays:

```
Comparison PASSES on ASCII number #.
Test continues.
```

If the character strings' difference and the GUI's **Quit on Number of Diff Errors** or the command line's **Xplabak5 -QQ** options are *not* specified, the status window displays:

```
Comparison FAILS on ASCII number #.
Test continues.
```

If the character strings' difference and the GUI's **Quit On Number of Diff Errors** or the command line's **Xplabak5 -QQ** options are specified and the number of errors is equal to or more than the number specified, the status window displays:

```
Comparison FAILS on ASCII number #
Test stopped before completion.
```

Please see Section 3.4.1.4 for detailed information on the *Quick Check* options.

7.5.3 Window Character Capturing — Save ASCII Window Button

To synchronize on a character string using the **Hotkey** window's **Save ASCII Window** button:

1. When you want to save the characters from a window, press the **F2** function key.

   The recording session pauses and the **Hotkey** window pops up.

   The status window states:

   ```
   Command Mode. Select option from menu.
   ```
2. Select the **Save ASCII Window** button.

   The screen pointer becomes a cross-hairs.
3. Move the cross-hairs to the window to be saved.
4. Click and release the mouse button, making sure not to click on an application's activation buttons.

   The status window states:

   ```
   ASCII window saving...
   ```

   The OCR technology will take some time to read all the values from the window so the system may hesitate for a while. When the characters are interpreted, the **OCR Verify** window pops up with all the window's values displayed.

   - Use the scroll bars to maneuver throughout the **OCR Verify** window.
   - Click on **OK** if the character string displayed is what you want.
   - Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
   - Click on **Retry** if the window you captured is not what you intended to capture. The status window states: `ASCII window saving...retry`. The screen pointer turns into a cross-hairs so you can capture another window's values.
   - If necessary, click on the **Edit** button to modify the window's captured values. When this button is selected it toggles to **Done** and the other buttons are dimmed. Move the mouse pointer to the location where you want to edit and click on the mouse button. A cursor will appear, indicating you can begin editing. Terminate an editing session by clicking on **Done**. This action will toggle the button back to **Edit** and the other options will become available.

5. If the **OK** button was selected, *CAPBAK/X* signals capture of the window's values with a bell and the status window states:

   ```
   ASCII window saving...complete.
   ```

   The characters are saved as `basename.xxx`, where `a` indicates that it is an ASCII character baseline file and `xxx` shows the original sequence in which the file was captured. You can view this file with any text editor.

   CAPBAK/X generates an `{cb_save_window_ascii ["test name", sequence,} [0 X Y width height]` event statement in the keysave file. Here, *X*, *Y* are the coordinates of the corner at which you started drawing the rectangle and *width* and *height* are the dimensions of that rectangle.

   The OCR technology also creates a temporary working file of the characters saved named `ocr.out`.

6. The system signals capture with a bell and the status window states:

   ```
   ASCII window saving...complete.
   ```

7. Use the **Hotkey** window as you normally would to resume recording or to issue other commands.

   During playback the `{cb_save_window_ascii ["test name", sequence,} [0 X Y width height]` statement is interpreted and a response file of the characters is captured named `basename.xxx`, in the RSP_a directory as an ASCII character response file, and `xxx` shows the original sequence in which the image was captured.

Playback verifies that *basename.xxx* was captured with a bell and the following message is in the status window of the **Record/Play** window:

```
ASCII window saving....complete.
```

If the *Quick Check* mode of playback is used with the GUI's **Quick Check Options** window or the command line's **Xplabak5 -QQ** option, *BSL_a/basename.xxx and RSP_a/basename.xxx* are automatically compared.

If the character strings compare with no difference, the status window displays:

```
Comparison PASSES on ASCII number #.
Test continues.
```

If the character strings' difference and the GUI's **Quit On Number of Diff Errors** or the command line's **Xplabak5 -QQ** options are *not* specified, the status window displays:

```
Comparison FAILS on ASCII number #.
Test continues.
```

If the character strings' difference and the GUI's **Quit On Number of Diff Errors** or the command line's **Xplabak5 -QQ** options are specified and the number of errors is equal to or more than the number specified, the status window displays:

```
Comparison FAILS on ASCII number #.
Test stopped before completion.
```

Please see Section 3.4.1.4 for complete information on *Quick Check* options.

7.5.4 Window Character Capturing — Using the Ctrl + F6 Keys

To capture the values from an area of the screen using the **Ctrl** and **F6** keys:

1. Move the mouse pointer so it is on top of the window containing the values you want to save.

   If you want to save a pull-down menu, click on the pull-down menu and drag the mouse pointer so that it is located within the menu button's borders. If the mouse pointer is located between two buttons or on a button's borders when the window is captured, recording may fail and the window may not be captured.

2. Press the **Ctrl** and **F6** keys at the same time and then lift them up. Unlike **Ctrl F5** which captures an area which you designate, **Ctrl F6** captures the entire active window.

   The OCR will take some time to read all the values from the window so the system may hesitate for awhile. When the characters are interpreted, the **OCR Verify** window pops up with all the window's values displayed.

   - Use the scroll bars to maneuver throughout the **OCR Verify** window.
   - Click on **OK** if the character string displayed is what you want.
   - Click on **Cancel** if you do not want to use the character string displayed and you want to issue another command.
   - Click on **Retry** if the window you captured is not what you intended to capture. The status window states:

     ```
     ASCII window saving...retry.
     ```

     The screen pointer turns into a cross-hairs so you can capture another window's values.
   - If necessary, click on the **Edit** button to modify the window's captured values. When this button is selected it toggles to **Done** and the other buttons are dimmed. Move the mouse pointer to the location where you want to edit and click on the mouse button. A cursor will appear, indicating you can begin editing. Terminate an editing session by clicking on **Done**. This action will toggle the button back to **Edit** and the other options will become available.

3. If the **OK** button was selected, *CAPBAK/X* signals capture of the window's character values with a bell and the status window states:

   ```
   ASCII window saving...complete.
   ```

   The characters are saved as *basename.xxx*, in the BSL_a directory an ASCII character baseline image and *xxx* shows the original sequence in which the file was captured. You can view this file with any text editor.

   CAPBAK/X generates a `{cb_save_window_ascii ["test name", sequence,`*0 X Y* *`width height`*`]` event statement in the keysave file. Here, *X*, *Y* are the coordinates of the corner at which you started drawing the rectangle and *`width`* and *`height`* are the dimensions of that rectangle.

   The OCR technology also creates a temporary working file of the characters saved named `ocr.out.`

4. Continue the recording session.

   During playback the `{cb_save_window_ascii ["test name", sequence,} [`*0 X Y* *`width height`*`]` statement is interpreted and a response file of the characters is captured to a `RSP_a` directory, which indicates that it is a response file; `xxx` shows the original sequence in which the file was captured. Playback verifies that *`basename`*`.xxx` was captured with the following message in the status window of the *CAPBAK/X* invocation window:

   ```
   ASCII window saving....complete.
   ```

If the character strings compare with no difference, the status window displays:

```
Comparison PASSES on ASCII number #.
Test continues.
```

If the character strings' difference and the GUI's **Quit On Number of Diff Errors** or the command line's **Xplabak5 -QQ** options are *not* specified, the status window displays:

```
Comparison FAILS on ASCII number #
Test continues.
```

If the character strings' difference and the GUI's **Quit On Number of Diff Errors** or the command line's **Xplabak5 -QQ** options are specified and the number of errors is equal to or more than the number specified, the status window displays:

```
Comparison FAILS on ASCII number #
Test stopped before completion.
```

Please see Section 3.4.1 of this guide for complete information on the *Quick Check* options available.

7.6 Multi-Platform OCR Font Training

In client / server applications for *CAPBAK/X* there are more and more situations in which connections are made between UNIX and Windows machines. For example, a UNIX-based client may be viewing material generated on a Windows NT based server. One such product which uses both fonts on the UNIX side is called "Causation" by UniPress.

The font training enhancement described below provide font training files that handle the most-common:

- PC Only Fonts
- UNIX Only Fonts
- Combined UNIX and PC Fonts

7.6.1 PC Fonts Recognized

PC-based font character recognition support is provided for the following standard Microsoft TrueType fonts:

Times Roman
Courier
MS-Line
MS-San Serif

Currently, training is optimized to recognize fonts at 8, 10, and 12 point sizes, including bold face. Our experience is that once PC-based text is emulated in a UNIX X-Windowing environment, the most legible size, due to the screen resolution difference between PC and UNIX hardware, is a 12 point PC-font. MS-Windows emulations using UNIX standard color map is supported.

7.6.2 UNIX Only Fonts Recognized

This varies from platform to platform. However, in all cases, the UNIX OCR training supports 95% of the most-commonly used typefaces.

7.6.3 Combined UNIX & PC Fonts Recognized

In addition to the standard set of UNIX-specific fonts supported, optical character recognition for PC-based fonts is available concurrently. This is helpful when the majority or mission-critical applications are developed in a UNIX environment yet the Application Under Test (AUT) resides in a MS-Windows environment.

Another scenario is if managing software resides in UNIX and has not been nor will be ported to Windows, yet the AUT is MS-Windows based. Supported PC-based fonts are listed in the proceeding section.

7.6.4 How to Choose the Right Training File, When to Use Which File

Optical character recognition training is based on a neural-network like structure. Though theoretically, an OCR training file can accept as many fonts as trainable, the law of statistics overtakes the results. This is why we chose not to train for every conceivable typeface in the market.

We offer PC-UNIX font recognition without significant compromise to the recognition success rate. Based on this premise, we encourage users to select the most appropriate training file while testing their AUT.

If the AUT is MS-Windows based exclusively, use the PC-only OCR training file. On the other hand, if testing requires the monitoring of both UNIX and PC activity, use the PC-UNIX based OCR training file.

7.7 RGB Specification Option

7.7.1 OCR Color Processing Enhancement Summary

OCR recognition effectiveness can be affected by *CAPBAK/X*'s built-in algorithms for processing closely-matched colors, and the effectiveness of *CAPBAK/X* in a realistic scenario when such images are used is much lower. Basically, the OCR engine becomes confused by the colors and the recognition percentage drops off sharply.

The enhancement described below gives you additional options in selecting pixel values or rgb values. This, in turn, greatly improves OCR efficiency.

7.7.2 Background and Technical Approach

7.7.2.1 How CAPBAK/X Currently Works, Why the New Facilities are Needed

Without using the options described below, *CAPBAK/X* currently captures images and dithers them into a 1-bit plane equivalent. The derived image with foreground / background is set according to the current definitions for white and black.

The problem is that the OCR engine fails if there is text embedded in complex images, e.g. red letters against a pink background. What is foreground and what is background affects which 1-bit plane derived image is sent to the OCR.

The difficulty is that the built-in dithering misleads the OCR engine; OCR tries to extract ASCII from a sloppy image.

7.7.2.2 The Technical Approach

User specifies either *rgb.rc*, a set of RGB color values and a threshold for applying them, or *pixel.rc*, a set of screen pixel values to be cast foreground / background. With this new capability you can override the built-ins if you want, and fully specify the rules by which the 1-bit-plane derived image is generated. This gives you nearly complete control over the OCR recognition process.

7.7.3 Special *pixel.rc* and/or *rgb.rc* Files

7.7.3.1 Where You Put These Files

You put either the *rgb.rc* or the *pixel.rc* file at *$SR/ocr*. If there is no *$SR/ocr* directory, or if *$SR/ocr* exists and does not contain either *rgb.rc* or *pixel.rc*, then *CAPBAK/X* acts normally using its own built-ins.

If *$SR/ocr/pixel.rc* exists, *pixel.rc* data is used even if *$SR/ocr/rgb.rc* exists.

If *$SR/ocr/rgb.rc* exists but *$SR/ocr/pixel.rc* does **not** exist, *CAPBAK/X* uses *$SR/ocr/rgb.rc* data.

7.7.3.2 Precedence Between *rgb.rc* and *pixel.rc*.

A *pixel.rc* file, if present, takes precedence over an *rgb.rc* file, if present. If there is no *pixel.rc* file present, then *CAPBAK/X* tries to find an *rgb.rc* file and if it finds one, it applies the values found there. If neither a *pixel.rc* file nor a *rgb.rc* file is found, *CAPBAK/X* behaves normally.

These two capabilities let you customize the *CAPBAK/X* OCR for maximum extraction of useful ASCII information from a captured screen image.

7.7.3.3 *rgb.rc* FILE FORMAT

The user specifies a set of up to 255 RGB values represented as follows in the *rgb.rc* file that is read during OCR activation time by *CAPBAK/X*:

7.7.3.4 Foreground RGB Specification Format

You specify the rgb color values you want to be treated as the foreground with the *rgb.rc* file using one line for each color value, as show below:

```
foreground
000 000 000
120 120 120
122 147 210
064 064 064
...
<EOF>
```

7.7.3.5 Reverse RGB Specification Format

If the *rgb.rc* file contains a different initial line, as shown below, and has the effect of specifying the colors that become the background (rather than the foreground):

```
background
255 255 255
120 120 120
255 200 195
...
<EOF>
```

then the process will be the same, except that the specified colors are those which are mapped into the background, all others being mapped into the foreground.

7.7.3.6 Tolerance Specification

You can, with the *rgb.rc* file and with either of the above two formats, specify the tolerance at which RGB values are matched between the current screen image and the baselined image. You do this by adding a tolerance phrase to the first line of the file:

```
background tolerance <value>
255 255 255
120 120 120
255 200 195
...
<EOF>
```

where <value> is a decimal number from 0 to 100 that specifies the percentage of closeness for the two RGB values to match.

7.7.3.7 Formatting Rules, *rgb.rc* File

This input file format follows that commonly used in the *rgb.txt* file (see below). Note that any information on any line in the file from the twelfth character and beyond is ignored. Any lines not matching the above format result in an error message and are ignored.

Note that you can choose any 24-bit (3 x 8-bit) RGB value that is possible on the machine in this fashion. Every color that is given is put in as a fore-ground color in the derived image that is generated from the captured image and sent to the OCR engine.

You have to have at least one valid color or the file is ignored.

The file must begin with either foreground or background (see below) or the file is ignored and operation reverts to normal mode.

7.7.3.8 RGB File Option

This extension to *CAPBAK/X* lets you specify a list of Red / Green / Blue (RGB) values that make up the foreground color [the background color], with any other colors being assigned to the background color [the foreground color]. The standard is to specify RGB values as three 8-bit quantities.

7.7.3.9 Pixel File Option

In addition, you can use a *pixel.rc* file and specify pixel values that make up the foreground color [the background color], with any other colors being assigned to the background color [the foreground color]. The standard is to specify Pixel values as a single 8-bit pixel, except for monochrome displays (in which case these options are unnecessary).

7.7.3.10 *rgb.rc* Operation Mode

The new RGB specification feature is active only in case there is a *rgb.rc* file present in the directory *$SR/ocr*. If this file is absent then *CAPBAK/X* behaves normally. Note that $SR is an environment variable that points to the directory at which *TestWorks* is installed.

7.7.3.11 *rgb.rc* File Absent

If the *rgb.rc* file is **not present** in the current working directory, then the OCR software behaves as it currently does.

7.7.3.12 Normal Operation

CAPBAK/X, when presented with a color image and asked to extract ASCII text from it using the built-in OCR engine, converts the image through a process called dithering into a 1-pixel plane deep monochromatic image derived from the screen image.

Normally this image contains derived pixels (actually, single bits) that are selected on the basis of known machine characteristics, properties of the color map settings, and several other factors. *CAPBAK/X* settings are preset to give effective use of the OCR capability on most machines, with most displays, and on most present platforms.

7.7.4 Operation with *rgb.rc* File Present

We have provided an algorithm that (described here in pseudocode) works in the following way. The candidate image is the one the user has selected on the screen, and the derived image pixels, with foreground and background as 1 and 0, respectively, are found by looking up the RGB values for the actual pixel to see if they are in the user-supplied *rgb.rc* file:

```
for (all pixels in the candidate image)
   {
   for (each defined color in the list in
   rgb.rc)
         if (color of pixel = color in list)
         {
               derived-image-pixel = foreground = 1;
               stop the scan;
         }
         No match was found, so:
         derived-image-pixel = background = 0;
   }
Activate the OCR engine using the derived image;
```

Note: Because the innermost loop stops if a color-match is found between some color specified in the *rgb.rc* file and the current pixel, it is best to put the color values in the *rgb.rc* file in the order that is most likely to match. For short lists of RGB values overall performance is not expected to be a problem.

This gives you the flexibility to specify a set of colors all of which are aggregated to be the foreground color.

7.7.4.1 How This Method Works

To make sure there is no confusion, but without requiring the reader understand all of the intricate details of how the UNIX X Window system works, it will be useful to point out how the above algorithm is actually implemented.

The X Window system actually renders each colored pixel from a machine dependent set of three 16-bit color intensity values. There are X Window calls that extract a C structure from the actual three 16-bit quantities that are used on your system and show you what values are actually being used to generate your display. This is done via the X Window call *XQueryColors(...)*;.

The three 8-bit rgb values that are specified in the table that a user specifies are converted into the same X Window structure with the function call *XParseColor.*

Hence, the comparison above (at the Noted line above) is done at this level, i.e. by comparing the outputs of these two function calls for the particular pixel in question.

Note that you specify the RGB values in your *.Xdefaults* file, or in dozens of different ways within your application. The most common source for this information is the named colors in the file *rgb.txt* which is normally found */usr/lib/X11/...*. The specific location will vary from system to system.

7.7.5 Operation with *pixel.rc* File Present

The new pixel value specification feature is active only in case there is a *pixel.rc* file present in the directory *$SR/ocr*. If this file is absent, and if there also is no *rgb.rc* file in *$SR/ocr*, then *CAPBAK/X* behaves normally.

Note: *$SR* is an environment variable that points to the directory at which TestWorks is installed.

7.7.5.1 *pixel.rc* File Format

The user specifies a set of up to 255 RGB values represented as follows in the *pixel.rc* file that is read during OCR activation time by *CAPBAK/X*:

```
foreground
00
12
12
06
FF
...
<EOF>
```

7.7.5.2 Reverse Pixel Specification Format

If the *pixel.rc* file contains a different initial line, as follows:

```
background
FF
36
25
...
<EOF>
```

then the process will be the same, except that the specified colors are those which are mapped into the background, all others being mapped into the foreground.

7.7.5.3 Formatting Rules, *pixel.rc* File

This input file format follows that commonly used in the *rgb.txt* file.

Note: any information on any line in the file from the fourth character and beyond is ignored.

Any lines not matching the above format generate an error message to standard output and are ignored.

Note: you can choose any 8-bit pixel value that is possible on the machine in this fashion. Every color that is given is put in as a foreground color in the derived image that is generated from the captured image and sent to the OCR engine.

You have to have at least one valid pixel specified or the file is ignored.

The file must begin with either foreground or background or the file is ignored and operation reverts to normal mode.

7.7.6 Operation with the *pixel.rc* File Present

Here is the processing algorithm for *pixel.rc* processing.

```
for (all pixels in the candidate image)
    {
    for (each pixel in the list from pixel.rc)
       if (actual pixel = pixel list)
       {
             derived-image-pixel = foreground = 1;
             stop the scan;
       }
       No match was found, so:
       derived-image-pixel = background = 0;
    }
Activate the OCR engine using the derived image;
```

7.7.7 Obtaining data for *pixel.rc* or *rgb.rc*

Where do you get the data for these files?

7.7.7.1 Obtaining *pixel.rc* Data

Use the utility xmag (supplied), which comes from the X11 samples and is freely distributable, to examine actual pixel values and 16-bit RGB values, but not the 8-bit RGB values used to program *rgb.rc*.

7.7.7.2 Obtaining *rgb.rc* Data

Use the values from *rgb.txt* at */usr/lib/X11...*

Get color values from *xmag* (supplied) or some equivalent utility.

CHAPTER 8

Xvirtual Display System

INFORMATION: Operation of **Xvirtual Display**. Capability to simulate multi-user sessions from one machine for purposes of load generation, performance assessment, and multiple-test execution. It can run tests in parallel to speed up execution of a test suite.
SEE ALSO: Chapter 1, "Introduction".
LEVEL: Intermediate to Advanced users.

8.1 Platform Availability

The **Xvirtual Display** system is available for these platforms:

- SPARC under Sun / OS running X11
- SPARC under Solaris running X11
- RS / 6000 under AIX 3.2.*n*
- SGI
- HP

The **Xvirtual Display** system is intended for software testers who are familiar with any type of capture / replay system, TrueTime or Object-Mode-based, that works with X11. Being a true and complete implementation of X11, **Xvirtual** will also work with any batch-mode application that can be launched with batch-type commands within an xterm. **Xvirtual** should be viewed as a backgroundable equivalent to a complete foreground X11 system in all regards.

In addition, you can use **Xvirtual** to test an application with different screen sizes and with screen attributes than the one on which **Xvirtual** has been launched.

8.2 Component Inventory

Below are the components of **Xvirtual Display**. The components should have been installed in the *install_dir/bin* directory during installation.

xvinit	This utility initializes **Xvirtual Display** and provides user-control over the execution-time properties of each virtual display. Please refer to another section for complete details (See Section 8.4 - "The xvinit Utility" on page 138.).
`xvinitrc`	This is a sample configuration file for the **Xvirtual** server. It has the same format and typical content as *xinitrc* file. Please refer to another section for a typical file sample (See Section 8.8.2 - "Using the .xvinitrc File" on page 146.).
Xvirtual	This is the modified version of the X11R5 server that acts as the server. Please refer to another section for option information (See Section 8.6 - "The Xvirtual Display Server" on page 141.).
xvmon	This is the virtual display server monitor. Please refer to another section for complete information (See Section 8.7 - "The xvmon Monitor/Client for Xvirtual" on page 142.).

8.2.1 Linking Manually

If you are not currently using X11, in order for the **Xvirtual** server to work, you must link SR's X11R5 library to your X11R5 library. To soft link manually:

1. Type

```
ln -s $SR/lib/X11R5 /usr/X11R5
```

This creates a soft link from the $SR/lib/X11R5 to /useX11R5.

2. You can modify the path so X11 Virtual Display System can be invoked from any directory. Set the user's path to include **`$SR/bin`** (i.e., the path to the executables).

8.3 General Operation with a Capture/Playback Tool

To run a test or a sequence of tests with the **Xvirtual Display** with a capture / playback tool:

1. Record a test as you normally would, entering keystrokes and mouse movements and saving images with the capture / playback system.
2. In order to use the **Xvirtual** server, you must be running the Motif 1.2 window manager (**mwm**).
3. When you are ready to play back on virtual displays, use the **xvinit** utility to initialize the modified virtual display server, **Xvirtual**. The **xvinit** utility is further discussed in another section (See Section 8.4 - "The xvinit Utility" on page 138.)
4. Use **xvmon** to select and display any virtual display. See another section for further information (See Section 8.7 - "The xvmon Monitor / Client for Xvirtual" on page 142.).
5. Use the standard invocation playback syntax for the capture / playback system you are using to play back the test script.

When using *CAPBAK/X*, a common playback command on a virtual display may be:

```
Xplabak5 -display display_name:1 -k base-
name.ksv -S
```

where **`Xplabak5`** is the command to play back a test script, **`-display`** specifies the display to which the test script is to be played back, **`:1`** is virtual display screen **1**, **`-k`** ***`basename.ksv`*** is the name of the test script to be played back, and **`-S`** is the switch to suppress start-up messages. For further examples of the **`Xplabak5`** command, please another section (See Section 4.4.1 - "Xplabak5 command" on page 74.).

6. Images captured images during your recording session with *CAPBAK/X*, playback should capture the corresponding response file images for each display.

If you play back the same session on more than one virtual display in the same directory, the response file images will overwrite each other. Use the **Xplabak5 -r** ***`response prefix`*** option for each virtual playback to avoid this. It will save all response images to a prefix other than ***`base-name`***. Make sure, of course, that each playback uses its own unique ***`prefix`***. For examples (See Section 4.4 - "Command-Line Invocation" on page 74.).

7. You can then compare the response file images of actual behavior with the baseline images of expected behavior. This can be done during playback using the *CAPBAK/X Quick Check* mode, or after playback with SR's **Xexdiff** utility (see the *EXDIFF User's Guide* for further information).

8.4 The xvinit Utility

You can control operation of each virtual display server, **Xvirtual**, entirely through the **xvinit** command. This utility can be used to set up the screen in any required way.

When you normally invoke the X Window System, you use the **xinit** utility to initialize the X11R5 server. Likewise, the **xvinit** utility initializes one copy of **Xvirtual** at the indicated display.

You start up a copy of the virtual display server, **Xvirtual**, by using the initialization utility **xvinit** as follows:

`xvinit [ [-r start-up script | X-client] options] [-- [server] [display] options]`

`xvinit`	The **`xvinit`** command is used to start the **Xvirtual** server and a first client program (usually a terminal emulator). When the first client exits, **`xvinit`** will kill the **Xvirtual** server and then terminate.

During start-up, if the **`-r`** switch is specified as the first option, then **xvinit** will read from the file specified. If **-r** is not specified, then **xvinit** will try to use the value from $XVINITRC, if it is set as an environment variable. If is not specified, **`xvinit`** will attempt to read from the *.xvinitrc* file in the user's home directory to run as a shell script to start up client programs. If no such file exists, **xvinit** executes the following default command line:

```
xvmon -geometry +1+1 -xvmondisplay :0
```

If no specific server is given on the command line, **`xvinit`** will use the value specified for environment variable $XSERVERRC, if it is set. If is not specified, **`xvinit`** will attempt to read from the *.xserverrc* file in the user's home directory. If no such file exists, **`xvinit`** will try to start the default **Xvirtual** server with the following default command line:

```
Xvirtual :1
```

Note: This command assumes that there is a program named **Xvirtual** in the current search path.

Programs that are run by *.xvinitrc* and *.xserverrc* should be run in the background if they do not exit right away, so that they do not prevent other programs from starting up. The last program started from these scripts, usually a window manager or **xvmon**, should be run in the foreground so that the script does not exit and cause **xvinit** to exit.

`-r` ***`start-up script`***

Specifies an alternative to the **xvinit** start-up script, `.xvinitrc`.

`X-client`	Specifies an alternate client and command line *options*. The default is `xvmon -geometry +1+1 -xvmondisplay:0`.
`options`	Specifies options for the X-client or **xvmon**.

`[-- [`*`server`*`] [`*`display`*`] `*`options`*`]`

Specifies an alternate X-server command line. If ***`server`*** is omitted, **xvinit** runs the default server (`Xvirtual: 1`). If *display* is omitted, **`xvinit`** runs the server on the default screen (`:0`). **`xvinit`** passes ***`options`*** as arguments to the server command.

Both the client program name and the server program name must begin with a slash "/" or a period ".". Otherwise, they are treated as arguments to be appended to their respective start-up lines. This makes it possible to add arguments (for example, foreground and background colors) without having to retype the whole command line.

8.4.1 Related Environment Variables

Below are related environment variables that are used in the **Xvirtual Display** environment.

DISPLAY	Specifies the default host and display number.
XVINITRC	Specifies an *init* file containing shell commands to start up the initial windows. By default *xvinitrc* in the home directory will be used.
XVSERVERRC	Specifies the actual display for the **xvmon** client.

8.4.2 Related Files

`.xvinitrc`	Default client script.
`.xvserverrc`	Default server script.

8.5 .xvinitrc File

Below is a typical .xvinitrc file:

```
xterm -C -T console &
mwm 2> /tmp/mwm.log &
xvmon
```

Note: **xvmon** is the last client specified in the file. It will initiate a window that allows the user to view each server it is connected to. Exiting **xvmon** will terminate the virtual display server, **Xvirtual**. See Section 8.6 for further information.

8.6 The Xvirtual Display Server

Below is the syntax for the virtual display server, **Xvirtual**:

Note: If you want each virtual display to match the size of the display you did the recording on, then specify the width, height and depth of the virtual display with the **`-width`**, **`-height`** and **`-depth`** options below.

`Xvirtual[`*`:displaynum`*`] [`*`x11r5options`*`...] [`*`options`*`]`

`Xvirtual`	Modified X11 display server (Release 5). It is most often started with the **xvinit** start-up script.
`:displaynum`	Sets the display number of the server. For example, `Xvirtual :1` allows clients with `DISPLAY=:1` to establish connections. The default display number is 1.
`x11r5options`	Specifies any of the standard server options for X11R5.
`options`	
`-width` *N*	Specifies the width of **Xvirtual**.
`-height` *N*	Specifies the height of **Xvirtual**.
`-depth` ***N***	Specifies the depth of **Xvirtual**.
`-dpi` *NxN*	Specifies the X and Y resolution of **Xvirtual**.
`-maxt N`	Specifies the maximum number of seconds **Xvirtual** runs before exiting.
`-q`	Redirects messages to *`/tmp/vrtlfile`*.

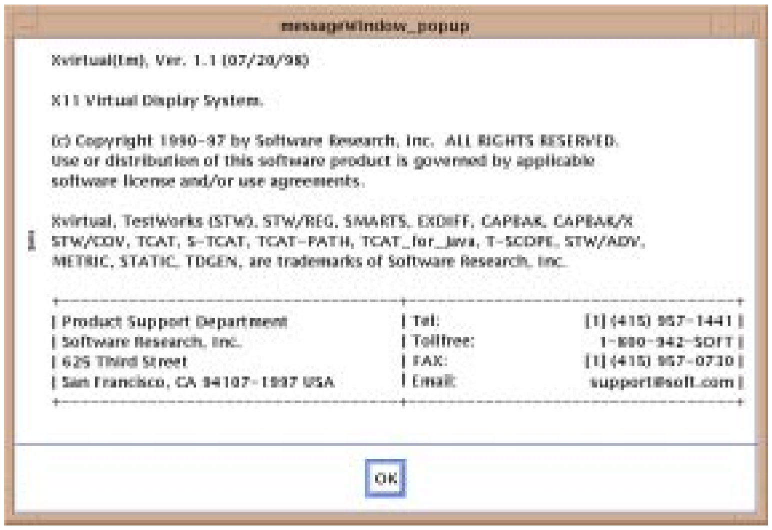
Note: In order to use the **Xvirtual** server, you must be running the Motif 1.2 window manager (**mwm**).

8.7 The xvmon Monitor/Client for Xvirtual

Below is the syntax for **xvmon**:

`xvmon [`*`toolkitoptions ...`*`] -xvmondisplay :`*`num`*

`xvmon`	The **Xvirtual** server's monitor and client.
`toolkitoptions`	The **xvmon** client accepts all of the standard X-Toolkit command line options.
`-xvmondisplay :`*`num`*	Specifies the actual display for **xvmon**. If this option is set, this option will overwrite the `XVMONDISPLAY` environment variable.
`-logo`	Invokes a copyright notice each time **xvmon** is invoked if supplied. If this option is *not* supplied, then **xvmon** will display the copyright notice only if the *.xvinitr*c file has not been read for more than six hours.



```
Xvirtual(tm), Ver. 1.1 (07/20/98)

X11 Virtual Display System.

(c) Copyright 1990-97 by Software Research, Inc.  ALL RIGHTS RESERVED.
Use or distribution of this software product is governed by applicable
software license and/or use agreements.

Xvirtual, TestWorks (STW), STW/REG, SMARTS, EXDIFF, CAPBAK, CAPBAK/X
STW/COV, TCAT, S-TCAT, TCAT-PATH, TCAT_for_Java, T-SCOPE, STW/ADV,
METRIC, STATIC, TDGEN, are trademarks of Software Research, Inc.

+-------------------------------------+-------------------------------------+
| Product Support Department          | Tel:                [1] (415) 957-1441 |
| Software Research, Inc.             | Tollfree:               1-800-942-SOFT |
| 625 Third Street                    | FAX:                [1] (415) 957-0730 |
| San Francisco, CA 94107-1997 USA    | Email:                support@soft.com |
+-------------------------------------+-------------------------------------+
```

FIGURE 51 Copyright Window

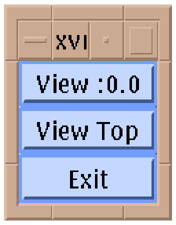


FIGURE 52 xvmon Window

This client is used to view and to exit the **Xvirtual** server. In other words, you can actually see what each virtual server is displaying. It allows you to monitor the tests that are in progress on the virtual displays.

It is comprised of the following buttons:

View : *virtual server #*

	Shows the entire virtual screen for virtual server #. The system will beep a few times and then it will take a few moments for the entire virtual screen to display.
View Top	Shows the topmost (active) window of the virtual display. This is useful for showing which client is active.
Exit	Terminates the **xvmon** client. If this is specified as the last client in the *.xvinitrc* file, it will also kill the **Xvirtual** server.

8.7.1 Related Environment Variable

Below is the environment variable that can be placed in the user's home directory.

`XVMONDISPLAY` Specifies the actual display for the **xvmon** client. Alternatively, you can use the **`-xvmondisplay`** option of **xvmon**.

8.8 Virtual Display Examples

Below are examples of how to use the **xvinit** utility, the `.xvinitrc` file and playback commands on a virtual display.

8.8.1 Using the xvinit Utility

Before using the **xvinit** utility, you have to set the environment variable `XVMONDISPLAY` to `:0` or the actual display.

`xvinit` Starts up the **Xvirtual** server and runs the user's *.xvinitrc*, if it exists, or else starts **xvmon**.

`xvinit --/usr/bin/X11/Xvirtual :1`

Starts up the **Xvirtual** server on a virtual display.

`xvinit -geometry =80x65+10+10 -fn 8x13 -j -fg white -bg navy`

Starts up a server named **Xvirtual**, and appends the given arguments to the default **xvmon** command. It ignores *.xvinitrc*.

`xvinit --/usr/bin/X11/Xvirtual :1`

Starts up the **Xvirtual** server on a virtual display.

`xvinit -fg white --/Xvirtual -l -c`

Uses the command **/Xvirtual -l -c** to start the server and appends the arguments **-fg white** to the default **xvmon** command.

`xvinit /usr/ucb/rsh fasthost cpupig -display ws:1 -- :1 -a 2 -t -5`

Starts the **Xvirtual** server on a virtual display with the arguments **-a 2 -t -5**. It then starts a remote shell on the machine **fasthost** in which it runs the command **cpupig**, telling it to display back on the local workstation.

`xvinit -- :1` Starts the **Xvirtual** server on virtual display #1, reading initial screen setup data from *.xvinitrc*.

`xvinit -r testB -- :2`

Starts the **Xvirtual** server on virtual display #2, reading initial screen setup data from *testB*.

`xvinit -r testB -- :3`

Starts the **Xvirtual** server on virtual display #3, reading initial screen setup data from *testB*.

```
xvinit -r testA -- :4
```

Starts the **Xvirtual** server on virtual display #4, reading initial screen setup data from *testA*.

8.8.2 Using the .xvinitrc File

Below is a sample `.xvinitrc` that starts a clock, several terminals, and leaves the xterm running as the "last" application. The user then chooses the **Exit** menu item to shut down **Xvirtual**.

```
xrdb -load $HOME/.Xresources
xsetroot -solid gray &
xclock -g 50x50-0+0 -bw 0 &
xload -g 50x50-50+0 -bw 0 &
xterm -g 80x24+0+0 &
xterm -g 80x24+0-0 &
mwm &
xvmon
```

Sites that want to create a common start-up environment could simply create a default *.xvinitrc* that references a site-wide start-up file:

```
 #!/bin/sh
. /usr/local/lib/site.xvinitrc
```

Another approach is to write a script that starts **xvinit** with a specific shell script. Such scripts are usually named *x11*, *xstart*, or *startx* and are a convenient way to provide a simple interface for novice users:

```
#!/bin/sh
xinit /usr/lib/X11/sys.startxrc -- /usr/
bin/X11/Xvirtual :1
```

8.8.3 Using A Playback Tool

Below are some samples of using the *CAPBAK/X* **`Xplabak5`** command to playback a test script on virtual displays. If you are using a capture / playback tool other than *CAPBAK/X*, substitute the **`Xplabak5`** command with your tool's commands:

To simulate multiple tests:

`Xplabak5 -display :1 -k script1`

Runs test script `script1` on virtual display #1.

`Xplabak5 -display :2 -k script2`

Runs test script `script2` on virtual display #2.

`Xplabak5 -display :3 -k script3`

Runs test script `script3` on virtual display #3.

To simulate load testing:

`Xplabak5 -d :2 -k script1 -r scripta &`

`Xplabak5 -d :1 -k script1 -r scriptb &`

Runs two of the same test on different virtual displays. To ensure captured response images (*`basename`*`.xxx`) don't overwrite each other, *`scripta`* and *`scriptb`* prefixes are set for each virtual playback.

8.9 Supplied Scripts

SR has created scripts to demonstrate how virtual displays work. You can customize these scripts to your own environment. The scripts are located in *`install_dir/demos/regression/Xvirtual.demo/passive`* and *`install_dir/demos/regression/Xvirtual.demo/play-back`*.

8.9.1 Passive Scripts

The scripts located in `/passive` demonstrate how virtual displays are set up and how they work in the background.

`Passive.Command`	Starts two X-client clocks.
`setup1v`	Sets up one **Xvirtual** server named **:1**, initiates one **xv-mon** virtual server monitor and initiates the *xvinitrc* file.
`setup3v`	Sets up three **Xvirtual** servers named **:1**, **:2** and **:3** and initiates a **xvmon** virtual server monitor and then initiates the `xvinitrc` file for each virtual server.

In addition to the scripts, we also supply:

`xvinitrc`	A start-up script file for the *setup1v* and *setup3v* scripts. This file configures the virtual server displays for **:1**, **:2** and **:3** and initiates the *Passive.Command* script.

8.9.1.1 Using the Passive Command Script

Passive.Command is used as part of the *xvinitrc* file to configure virtual displays. It starts two X-client clocks at the geometry coordinates +120,+120 and +320,+320.

```
# ---------------------------------------
---------------
# Passive.Command -- Script to execute an
Xvirtual instance.
# ---------------------------------------
---------------
# (c) Copyright 1994 by Software Research,
Inc. ALL RIGHTS RESERVED.
#
# The Passive.Command script is invoked as
part of the “xvinitrc”
# file. This command is given in the form
“xterm ...-e Passive.Command”
# in the “xvinitrc” file.
#
# The script invokes two instances of the
xclock on the Xvirtual
# screen; you can view the result with the
“View :<virtual_server_#>”
# button on xvmon.
# ---------------------------------------
---------------

sleep 1
xclock -geometry +120+120 -update 1&
sleep 1
xclock -geometry +320+320 -update 1&

sh
```

8.9.1.2 Configuring Virtual Displays

xvinitrc sets up a virtual display:

```
# ---------------------------------------
-----------------------
# xvinitrc -- Initialization file for the
Xvirtual server.
# ---------------------------------------
-----------------------
# (c) Copyright 1994 by Software Research,
Inc.  ALL RIGHTS RESERVED.
#
# The commands in “xvinitrc” are called by
each running copy of
# Xvirtual at its own start-up time. This
script’s commands configure
# the screen on the virtual display.
#
# This version runs the command “Pas-
sive.Command” to illustrate passive
# use of the Xvirtual Display system.
# ---------------------------------------
-----------------------

mwm > /tmp/mwm.log 2> /tmp/mwm.log &
xterm -geometry +20+20 -e “Passive.Com-
mand” &
exec xvmon
```

This file is called by the **xvinit** command in the *setup1v* and *setup3v* script to configure the server. This file starts the **mwm** window manager, invokes an xterm located at geometry coordinates +20+20, calls the *Passive.Command* script to invoke two X-client clock, and executes the **xvmon** monitor.

8.9.1.3 Using the setup1v Script to Set Up One Virtual Display

setup1v uses the *xvinitrc* file to create a **Xvirtual** display:

```
#!/bin/sh
# -------------------------------------------------------------------
# setup1v -- Script to set up one Xvirtual display.
# -------------------------------------------------------------------
# (c) Copyright 1994 by Software Research, Inc.  ALL RIGHTS RESERVED.
#
# Shell script to set up a single copy of Xvirtual in the background.
#
# The assignment of XVMONDISPLAY to the foreground display makes it
# possible to see the xvmon client. There is no limit on the number
# of copies of Xvirtual that you can run.
#
# The width and height of the display should match that of the current
# display.  You can find those facts out with "xdpyinfo | more".
# -------------------------------------------------------------------

XVMONDISPLAY=:0.0
export XVMONDISPLAY

xvinit -r xvinitrc -- :1 -width 1280 -height 1024 -dpi 91x92 &
xset s off -display :1
```

The **xvmon** monitor for the virtual display is at the current display (**:0**). You can use the monitor to look at the virtual display's top window (which is currently the X-client calculator at 320, 320 initialized with the *Passive.Command* script) or the entire display (See Section 8.9.1 - "Passive Scripts" on page 148.). Exit **xvmon** when you finish viewing the virtual display.

To set-up a **Xvirtual** server for display **:1**, the **xvinit** command uses the *xvinitrc* file, the *Xvirtual.sh* and **Xvirtual** options (See Section 8.6 - "The Xvirtual Display Server" on page 141.).

8.9.1.4 Using the setup3v Script to Set Up Three Virtual Displays

setup3v uses the *xvinitrc* file to create a **Xvirtual** display:

```
# !/bin/sh
# -----------------------------------------------------
--
# setup3v -- Script to set up three Xvirtual displays.
# -----------------------------------------------------
--
# (c)Copyright 1994 by Software Research, Inc.ALL RIGHTS
RESERVED.
#
# Shell script to set up multiple copies of Xvirtual in
the background.
#
# The assignment of XVMONDISPLAY to the foreground dis-
play makes it
# possible to see the xvmon client. There is no limit on
the number
# of copies of Xvirtual that you can run.
#
# The width and height of the display should match that
of the current
# display. You can find those facts out with “xdpyinfo |
more”.
# -----------------------------------------------------
--

XVMONDISPLAY=:0
export XVMONDISPLAY

xvinit -r xvinitrc -- :1 -width 1280 -height 1024 -dpi
91x92 &
xset s off -display :1
sleep 1
xvinit -r xvinitrc -- :2 -width 1280 -height 1024 -dpi
91x92 &
xset s off -display :2
sleep 1
xvinit -r xvinitrc -- :3 -width 1280 -height 1024 -dpi
91x92 &
xset s off -display :3
```

The `sleep 1` command ensures there is not an overload by waiting one second after each virtual display is initiated.

The three **xvmon**s monitor the three virtual displays at the current display (**:0**). You can use each monitor to view each virtual display's top window (which is currently the xterm) or the entire display (see Section 8.7). Exit each monitor when you have completed viewing virtual display.

8.9.2 Playback Scripts

The scripts located in */playback* demonstrate how virtual displays are set up and how to play back a script created from *CAPBAK/X*. If you do not have *CAPBAK/X*, then the supplied scripts will *not* work.

`go1`	Plays back a keysave file named *simple.ksv* on the current display.
`setup1v`	Sets up one virtual display server (**Xvirtual**) named **:1** and initiates one **xvmon** virtual server monitor.
`go1v`	Plays back *simple.ksv* on a **Xvirtual** named **:1**.
`setup3v`	Sets up three **Xvirtual** servers named **:1**, **:2** and **:3** and initiates a **xvmon** virtual server monitor for each virtual server.
`go3v`	Plays back *simple.ksv* on server displays **:1**, **:2** and **:3**.

In addition to the scripts, we also supply:

`simple.ksv`	Pre-recorded test script that can be played back. *simple.ksv* is located in the */playback* directory.
`mtimer`	Pre-compiled **mtime** utility. This utility indicates how long a script takes to play back on a display in milliseconds. *mtimer* is located in *install_dir/bin*.
`mtimer.c`	Source code file for the **mtimer** utility. The code is located in *install_dir/source*.
`xvinitrc`	A start-up script file for the *setup1v* and *setup3v* scripts. This file configures the virtual server displays for **:1**, **:2** and **:3**. *.xvinitrc* is located in *install_dir/bin*.

The following sections show you step-by-step how SR implemented the above scripts and files, providing you with a template.

8.9.2.1 Creating simple.ksv

As mentioned in the previous section, the supplied scripts use a keysave file named `simple.ksv`. `simple.ksv` was created with *CAPBAK/X* in the following way:

1. Prior to recording, two xterms were initialized: One in the lower left of the screen and one in the upper left of the screen (at geometry +20,+20)
2. The recording was initialized from the lower left xterm with *CAPBAK/X* **`Xcapbak5`** command.
3. Once the recording was started, the mouse was moved to the upper left xterm.
4. The xterm was activated with a mouse click, and **`date`** was typed in. The system returned with the current date.
5. **`time sleep 0`** was then entered to time how long the system takes to sleep 0 seconds. The result on most systems is close to 0.0 seconds.
6. **`xclock`** was then typed in to invoke the X client clock.
7. When the clock appears, it is dragged to a different location.
8. The clock ‘s upper left hand button was then clicked on. When the window menu appeared, **Close** was selected.
9. The recording session was then terminated.

8.9.2.2 Using the go1 Script to Playback

You can play back *simple.ksv* on your current server using the *go1* script. In order for the script to successfully execute, you must have an xterm located at +20,+20 on the screen.

```
# !/bin/sh
#-------------------------------------------
----------------
# go1 -- Script to play back one test ses-
sion on the current display.
# ------------------------------------------
----------------
# (c) Copyright 1994 by Software Research,
Inc.  ALL RIGHTS RESERVED.
#
# Shell script to playback “simple.ksv” in
the foreground.
# ------------------------------------------
----------------

mtimer Xplabak5 -q       -k simple.ksv -S
>> time1
```

mtimer outputs the millisecond time the keysave file took to execute to a file named *time1*.

8.9.2.3 Configuring Virtual Displays

xvinitrc sets up a virtual display:

```
# ---------------------------------------
---------------
# xvinitrc -- Initialization file for the
Xvirtual server.
# ---------------------------------------
---------------
# (c) Copyright 1994 by Software Research,
Inc. ALL RIGHTS RESERVED.
#
# The commands in “xvinitrc” are called by
each running copy of
# Xvirtual at its own start-up time. This
script’s commands configure
# the screen on the virtual display.
#
# This version runs the command “Pas-
sive.Command” to illustrate passive
# use of the Xvirtual Display system.
# ---------------------------------------
---------------

mwm > /tmp/mwm.log 2> /tmp/mwm.log &
xterm -geometry +20+20 &
exec xvmon
```

This file is called by the **xvinit** command in the `setup1v` and `setup3v` script to configure the server. This file starts the **mwm** window manager, invokes an xterm located at geometry coordinates +20+20 and executes the **xvmon** monitor.

8.9.2.4 Using the setup1v Script to Set Up One Virtual Display

setup1v creates a virtual display:

```
# !/bin/sh
# -------------------------------------------
------------
# setup1v -- Script to set up one Xvirtual dis-
play.
# -------------------------------------------
------------
# (c) Copyright 1994 by Software Research, Inc.
ALL RIGHTS RESERVED.
#
# Shell script to set up a single copy of Xvir-
tual in the background.
#
# The assignment of XVMONDISPLAY to the fore-
ground display makes it
# possible to see the xvmon client. There is no
limit on the number
# of copies of Xvirtual that you can run.
#
# The width and height of the display should
match that of the current
# display. You can find those facts out with
“xdpyinfo | more”.
# -------------------------------------------
-----------XVMONDISPLAY=:0
export XVMONDISPLAY

xvinit -r xvinitrc -- :1 -width 1280 -height
1024 -dpi 91x92 &
xset s off -display :1
```

The **xvmon** monitor for the virtual display is at the current display (**:0**). You can use the monitor to look at the virtual display's top window (which is currently the xterm) or the entire display (See Section 8.7 - “The xvmon Monitor / Client for Xvirtual” on page 142.). Do not exit this window, as it is necessary to run the `go1v` script (See Section 8.9.2.5 - “Using the go1v Script to Playback on :1” on page 158.).

To set up a **Xvirtual** server for display **:1**, the **xvinit** command uses the `xvinitrc` file SR set up (See Section 8.9.2.3 - “Configuring Virtual Displays” on page 156.) and uses **Xvirtual** options (See Section 8.6 - “The Xvirtual Display Server” on page 141.).

8.9.2.5 Using the go1v Script to Playback on :1

If you used `setup1v` to set up a virtual display, you can automatically execute *simple.ksv* on that display with the `go1v` script:

```
          # ---------------------------------------
          ---------------
  # go1v -- Script to play back one test session on one Xvirtual display.
  # ---------------------------------------------------------------------
  # (c) Copyright 1994 by Software Research, Inc. ALL RIGHTS RESERVED.
  #
  # Shell script to playback “simple.ksv” in the background on display
  # :1.
  # ---------------------------------------------------------------------

          mtimer Xplabak5 -q -d :1 -k simple.ksv -S
          >> time1v
```

mtimer outputs the millisecond time the keysave file took to execute to a file named `time1v`.

While the script is executing, you can use the **xvmon** monitor on the current display (**:0**) to view the top window or the entire screen of the virtual display (**:1**).

8.9.2.6 Using the setup3v Script to Set-Up Three Virtual Displays

`setup3v` creates three virtual displays:

```
# !/bin/sh
# ----------------------------------------------------------------------
# setup3v -- Script to set up three Xvirtual displays.
# ----------------------------------------------------------------------
# (c) Copyright 1994 by Software Research, Inc. ALL RIGHTS RESERVED.
#
# Shell script to set up multiple copies of Xvirtual in the background.
#
# The assignment of XVMONDISPLAY to the foreground display makes it
# possible to see the xvmon client. There is no limit on the number
# of copies of Xvirtual that you can run.
#
# The width and height of the display should match that of the current
# display. You can find those facts out with “xdpyinfo | more”.
# ----------------------------------------------------------------------

XVMONDISPLAY=:0
export XVMONDISPLAY

xvinit -r xvinitrc -- :1 -width 1280 -height 1024 -dpi 91x92 &
xset s off -display :1
sleep 1
xvinit -r xvinitrc -- :2 -width 1280 -height 1024 -dpi 91x92 &
xset s off -display :2
sleep 1
xvinit -r xvinitrc -- :3 -width 1280 -height 1024 -dpi 91x92 &
xset s off -display :3
```

The three **xvmon** monitors the three virtual displays at the current display (**:0**). You can use each monitor to view each virtual display's top window (which is currently the xterm) or the entire display (See Section 8.7 - “The xvmon Monitor / Client for Xvirtual” on page 142.). Do not exit any of the monitors, as they are necessary to run the `go3v` script (See Section 8.9.2.7 - “Using the go3v Script to Playback on :1, :2, and :3” on page 160.).

8.9.2.7 Using the go3v Script to Playback on :1, :2, and :3

To simultaneously execute `simple.ksv` on **Xvirtual** server **:1**, **:2**, **:3**, use the *go3v* script. You can only use `go3v` if you used `setup3v` to create three virtual displays.

```
# ----------------------------------------------------------------------
# go3v -- Script to play back one test session on three Xvirtual
# display simultaneously.
# ----------------------------------------------------------------------
# (c) Copyright 1994 by Software Research, Inc. ALL RIGHTS RESERVED.
#
# Shell script to playback “simple.ksv” in the background on display
# :1, :2 and :3.
# ----------------------------------------------------------------------

mtimer Xplabak5 -q -d :1 -k simple.ksv -S  >> timev1 &
sleep 1
mtimer Xplabak5 -q -d :2 -k simple.ksv -S  >> timev2 &
sleep 1
mtimer Xplabak5 -q -d :3 -k simple.ksv -S  >> timev3 &
wait
```

`mtimer` outputs the millisecond time the keysave file took to execute to files named *timev1* (**:1**), *timev2* (**:2**) and *timev3* (**:3**).

The **`wait`** command waits for all three playbacks to complete before returning control to the prompt.

While the script is executing, you can use the **xvmon** monitor on the current display (**:0**) to view the top window or the entire screen of the virtual display (**:1**).

APPENDIX A

Customizing *CAPBAK/X*

This appendix explains where the resource setup information is stored and gives you instructions on how to change it.

A.1 *CAPBAK/X* Setup Information

You can customize the *CAPBAK/X* graphical user interface (GUI) and command-line options by modifying the X Window System resource or setup files. This appendix explains where the setup information is stored and provides modification instructions.

Resource / setup files are text files, which can be modified with any UNIX text editor. All of the GUI defaults and some of the command-line defaults are set in the *SR* file supplied with the product. During installation, the *SR* file is automatically copied to */usr/lib/X11/app-defaults* directory. For further information on the *SR* file, please refer to the *Installation Instructions* in the supplied *OPEN THIS FIRST!* envelope.

Note: If you have *CAPBAK/X* running OpenWindows, Version 3, the location of the *SR* file is *$OPENWINHOME/lib/app-defaults*

Following is a list of the defaults. By manually changing the SR file you can change the pre-set defaults and avoid resetting user interface parameters every time you want to record a session.

```
capbakX5.geometry: 500x400+0+0

capbakX5*File_pulldown.New.labelString: New...
capbakX5*File_pulldown.New.accelerator: Ctrl<Key>N
capbakX5*File_pulldown.New.acceleratorText: Ctrl+N
capbakX5*File_pulldown.Load.labelString: Load...
capbakX5*File_pulldown.Load.accelerator: Ctrl<Key>L
capbakX5*File_pulldown.Load.acceleratorText: Ctrl+L
capbakX5*File_pulldown.Save.labelString: Save
capbakX5*File_pulldown.Save.accelerator: Ctrl<Key>S
capbakX5*File_pulldown.Save.acceleratorText: Ctrl+S
capbakX5*File_pulldown.Set Directory.labelString: Set Direc-
tory...
capbakX5*File_pulldown.Set Directory.accelerator: Ctrl<Key>D
capbakX5*File_pulldown.Set Directory.acceleratorText: Ctrl+D
capbakX5*File_pulldown.Exit.accelerator: Ctrl<Key>C
capbakX5*File_pulldown.Exit.acceleratorText: Ctrl+C

capbakX5*Utilities_pulldown.Image Utilities.labelString: Image
Utilities...
capbakX5*Utilities_pulldown.Image Util.accelerator: Ctrl<Key>I
capbakX5*Utilities_pulldown.Image Util.acceleratorText: Ctrl+I

capbakX5*load.dirMask: *.ksv
capbakX5*messageSave.dirMask: *.msg

capbakX5*functionKey*commandkey.value: F2
capbakX5*functionKey*synckey.value: F3
capbakX5*functionKey*syncmousekey.value: F4
capbakX5*functionKey*partialkey.value: F5
capbakX5*functionKey*mousekey.value: F6
capbakX5*functionKey*rootkey.value: F7
capbakX5*functionKey*asciisynckey.value: control+F3
capbakX5*functionKey*asciipartialkey.value: control+F5
capbakX5*functionKey*asciimousekey.value: control+F6
capbakX5*functionKey*increasekey.value: F4
capbakX5*functionKey*decreasekey.value: F5
capbakX5*functionKey*pauseresumekey.value: F8
capbakX5*functionKey*endkey.value: F9
capbakX5*functionKey*enterexitobjectkey.value: ctrl+F2

capbakX5*option*outputSyncSwitch.set: True
capbakX5*option*outputSyncDelay.value: 30
capbakX5*option*outputSyncWinName.set: True
capbakX5*option*outputSyncQuit.set: True
```

```
capbakX5*option*outputSyncMove.set: True
capbakX5*outputSyncXOffset: 0
capbakX5*option*outputSyncXOffset.value: 0
capbakX5*outputSyncYOffset: 0
capbakX5*option*outputSyncYOffset.value: 0
!capbakX5*outputSyncXOffset: 8
!capbakX5*option*outputSyncXOffset.value: 8
!capbakX5*outputSyncYOffset: 24
!capbakX5*option*outputSyncYOffset.value: 24
capbakX5*option*imageSyncSwitch.set:True
capbakX5*option*imageSyncDelay.value: 30
capbakX5*option*imageSyncIntervalSwitch.set: True
capbakX5*option*imageSyncIntervalDelay.value: 3
capbakX5*option*imageSyncQuit.set: True
capbakX5*option*imageSyncMove.set: False
capbakX5*option*asciiSyncSwitch.set: True
capbakX5*option*asciiSyncSwitch.set: True
capbakX5*option*asciiSyncDelay.value: 30
capbakX5*option*asciiSyncIntervalSwitch.set: True
capbakX5*option*asciiSyncIntervalDelay.value: 3
capbakX5*option*asciiSyncQuit.set: True
capbakX5*option*asciiSyncMovePointer.set: True
capbakX5*option*differenceSwitch.set: False
capbakX5*option*differenceReportSwitch.set: False
capbakX5*option*differenceReportFile.value: Report
capbakX5*option*differenceQuit.set: False
capbakX5*option*differenceQuitNumber.value: 1
capbakX5*option*differenceSave.set: False
capbakX5*option*includeFrameSwitch.set: False
capbakX5*option*compressImageSwitch.set: False
capbakX5*option*compressCmd.set: False
capbakX5*option*compressCmdName.value: compress -f
capbakX5*option*uncompressCmd.set: False
capbakX5*option*uncompressCmdName.value: uncompress
capbakX5*option*hotkeyPositionSwitch.set: False
capbakX5*option*xposition.value: 0
capbakX5*option*yposition.value: 0
capbakX5*option*delaySwitch.set: False
capbakX5*option*delay.value: 1.0
capbakX5*option*eventSyncSwitch.set: True
capbakX5*option*eventSyncDelay.value: 30
capbakX5*option*rebaselineSwitch.set: False
capbakX5*option*confirmRebaselineSwitch.set: False
capbakX5*option*startExtensionType.value: AUTO

capbakX5*widgetModeOption*Play Delay.set: True
capbakX5*widgetModeOption*Play Delay.labelString: Play Delay
(msec)
capbakX5*widgetModeOption*Retry Delay.set: True
```

```
capbakX5*widgetModeOption*Retry Delay.labelString: Retry Delay
(msec)
capbakX5*widgetModeOption*Object Timeout.set: True
capbakX5*widgetModeOption*Object Timeout.labelString: Object
Timeout (msec)
capbakX5*widgetModeOption*Window Timeout.set: True
capbakX5*widgetModeOption*Window Timeout.labelString: Window
Timeout (msec)
capbakX5*widgetModeOption*Play Text.value: 1000
capbakX5*widgetModeOption*Retry Text.value: 500
capbakX5*widgetModeOption*Object Text.value: 2000
capbakX5*widgetModeOption*Window Text.value: 5000

capbakX5*scriptAreaLabel.labelString: Script Area
!capbakX5*scriptAreaFrame.paneMaximum: 300

capbakX5*messageAreaLabel.labelString: Message Area
capbakX5*messageAreaFrame.paneMaximum: 150

capbakX5*option*ocrForegroundColor.value: white,black,red,green
```

FIGURE 53 SR File

A.2 Defaults

The following sub-sections provide descriptions of the default switches.

When opening the GUI-version of *CAPBAK/X*, the on / off status of options and the default values of options are determined by the default switch setting in the resource file. Within the resource file, switch settings are indicated by `set`, with `False` interpreted as off, `True` interpreted as on and values indicated by `value:` *#*.

For further information on the options affected by the GUI defaults (See CHAPTER 3 - "Options Menu" on page 48.).

A.2.1 Default Directory and Directory Mask Settings

When naming files, most users prefer to name certain types of files with unique extensions, such as naming all keysave files, *basename.ksv*. When opening files, *CAPBAK/X* reads the resource file for default masks to filter directories (indicated by `directory`) and files which match the masks (indicated by `dirMask`), thereby expediting the process of locating a particular file type.

Note: If no path is specified, the mask defaults to the current working directory.

```
capbakX5*load.dirMask: *.ksv
```

Sets the keysave file directory mask to ***.ksv** for the **Load** submenu (under **File** pull-down) in the *CAPBAK/X* window.

Previous sections give instructions on using directory masks. (See Section 3.1.1 - “File Selection Windows” on page 37.), and for further information on the file selection windows for the *CAPBAK/X* window (See Section 3.1.2 - “Using a File Selection Window” on page 39.).

A.2.2 Output Sync Options Settings

These entries determine the defaults for the **Output Sync Options** window.

```
capbakX5*option*outputSyncSwitch.set: True
```

Defaults the **Output Sync** option to on.

```
capbakX5*option*outputSyncDelay.value: 30
```

Defaults the **Output Sync** option’s delay to 30 seconds.

```
capbakX5*option*outputSyncWinName.set: True
```

Defaults the **Do Not Use Window Name On Output Sync** option to off.

```
capbakX5*option*outputSyncQuit.set: True
```

Defaults the **Quit On Output Sync Error** option to off.

```
capbakX5*option*outputSyncMove.set: True
```

Defaults the **Displace Window On Output Sync** option to off.

These entries determine the defaults for the **Image Sync Options** window.

```
capbakX5*option*imageSyncSwitch.set: True
```

Defaults the **Image Sync** option to on.

```
capbakX5*option*imageSyncDelay.value: 30
```

Defaults the **Image Sync** option’s delay to 30 seconds.

```
capbakX5*option*imageSyncIntervalSwitch.set: True
```

Defaults the **Image Sync Interval** option to on.

```
capbakX5*option*imageSyncIntervalDelay.value: 3
```

Defaults the **Image Sync Interval** option’s interval to 3 seconds.

```
capbakX5*option*imageSyncQuit.set: True
```

Defaults the **Quit On Image Sync Error** option to off.

```
capbakX5*option*imageSyncMove.set: False
```

Defaults the **Displace Window On Image Sync** option to off.

A.2.3 ASCII Sync Options Settings

These entries determine the defaults for the ASCII Sync Options window:

```
capbakX5*option*asciiSyncSwitch.set: True
```

Defaults the **ASCII Synchronization** option to on.

```
capbakX5*option*asciiSyncSwitch.set: True
capbakX5*option*asciiSyncDelay.value: 30
```

Defaults the **ASCII Sync Delay** time to 30 seconds.

```
capbakX5*option*asciiSyncIntervalSwitch.set: True
```

Defaults the **ASCII Sync Interval** option to on.

```
capbakX5*option*asciiSyncIntervalDelay.value: 3
```

Defaults the **ASCII Sync Interval** time to 3 seconds.

```
capbakX5*option*asciiSyncQuit.set: True
```

Defaults the **Quit on ASCII Sync Error** option to on.

```
capbakX5*option*asciiSyncMovePointer.set: True
```

Defaults the **Relocate Pointer on ASCII Sync** option on.

A.2.4 Quick Check Options Settings

These entries determine the defaults for the **Quick Check Options** window.

```
capbakX5*option*differenceSwitch.set: False
```

Defaults the **Difference Images with Baseline** option to off.

```
capbakX5*option*differenceReportSwitch.set: False
```

Defaults the **Difference Report** option to off.

```
capbakX5*option*differenceReportFile.value: Report
```

Defaults difference report to **Report**.

```
capbakX5*option*differenceQuit.set: False
```

Defaults the **Quit On Number Of Diff Errors** option to off.

```
capbakX5*option*differenceQuitNumber.value: 1
```

Defaults the number of errors for the **Quit On Number Of Diff Errors** option to 1.

```
capbakX5*option*differenceSave.set: False
```

Defaults the **Save All Difference Images** option to off.

A.2.5 Other Options Settings

These entries determine the defaults for the **Other Options** window.

`capbakX5*option*includeFrameSwitch.set: False`

Defaults the **Include Window Frame** option to off.

`capbakX5*option*compressImageSwitch.set: False`

Defaults the **Compress Images** option to off.

`capbakX5*option*compressCmd.set: False`

Defaults the **Compress Command** option to off.

`capbakX5*option*compressCmdName.value: compress -f`

Defaults the **Compress Command** command to **compress -f**.

`capbakX5*option*compressCmd.set: False`

Defaults the **Uncompress Command** option to off.

`capbakX5*option*uncompressCmdName.value: uncompress`

Defaults the **Uncompress Command** command to **uncompress**.

`capbakX5*option*hotkeyPositionSwitch.set: False`

Defaults the **Hotkey Win Position** option to off.

`capbakX5*option*xposition.value: 0`

Defaults the **Hotkey Win Position X** coordinate position to 0.

`capbakX5*option*yposition.value: 0`

Defaults the **Hotkey Win Position Y** coordinate position to 0.

`capbakX5*option*delaySwitch.set: False`

Defaults the **Delay Multiplier** option to off.

`capbakX5*option*delay.value: 1.0`

Defaults the **Delay Multiplier** option's speed to 1.0.

`capbakX5*option*eventSyncSwitch.set: True`

Defaults the **Event Sync** option to on.

`capbakX5*option*eventSyncDelay.value: 30`

Defaults the **Event Sync** option's delay to 30 milliseconds.

`capbakX5*option*rebaselineSwitch.set: False`

Defaults the **Regenerate Baseline Images** option to off.

```
capbakX5*option*confirmrebaselineSwitch.set: False
```

Defaults the **Verify Images On Play Back** option to off.

A.2.6 Function Keys Settings

These entries determine the defaults for the **Function Keys** window.

`capbakX5*functionKey*commandkey.value: F2`

Sets the **Hotkey Popup** function key to **F2**.

`capbakX5*functionKey*synckey.value: F3`

Sets the **Save Partial Image** function key to **F3**.

`capbakX5*functionKey*syncmousekey.value: F4`

Sets the **Save Mouse Image** function key to **F4**.

`capbakX5*functionKey*partialkey.value: F5`

Sets the **Save Partial Image** function key to **F5**.

`capbakX5*functionKey*mousekey.value: F6`

Sets the **Save Mouse Image** function key to **F6**.

`capbakX5*functionKey*rootkey.value: F7`

Sets the **Save Root Image** function key to **F7**.

`capbakX5*functionKey*increasekey.value: F4`

Sets the **Speed Up** function key to **F4**.

`capbakX5*functionKey*decreasekey.value: F5`

Sets the **Slow Down** function key to **F5**.

`capbakX5*functionKey*pauseresumekey.value: F8`

Sets the **Pause/Resume** function key to **F8** (toggle).

`capbakX5*functionKey*endkey.value: F9`

Sets the **End Session** function key to **F9**.

`capbakX5*functionKey*asciisynckey.value: control+F3`

Sets the ASCII sync key sequence to **CTRL** + **F3**.

`capbakX5*functionKey*asciipartialkey.value: control+F5`

Sets the ASCII Partial key sequence to **CTRL + F5**.

`capbakX5*functionKey*asciimousekey.value: control+F6`

Sets the Capture ASCII window key sequence to **CTRL + F6**.

```
capbakX5*functionKey*enterexitwidgetkey.value:
               control+F2
```

Sets the key sequence for entering and exiting Object-Mode recording the **CTRL + F2** (toggle).

A.2.7 Recording Options

```
capbakX5*widgetModeOption*Window Timeout.set:True
capbakX5*widgetModeOption*Window  Timeout.labelString:
               Window Timeout (msec)
capbakX5*widgetModeOption*Play Text.value: 1000
```

Sets **Play Delay** option default to 1 second.

```
capbakX5*widgetModeOption*Retry Text.value: 500
```

Sets **Retry Delay** option default to 5 seconds.

```
capbakX5*widgetModeOption*Object Text.value: 2000
```

Sets **Object Timeout** option default to 2 seconds.

```
capbakX5*widgetModeOption*Window Text.value: 5000
```

Sets **Window Timeout** option default to 5 seconds.

```
!capbakX5*widgetModeOption.OK.labelString: Apply
capbakX5*option*startextension: OOEXT
```

Sets *CAPBAK/X* recording to begin in ObjectMode.

```
capbakX5*option*startextension:XTRAPEXT
```

To start TrueTime recording using Xtrap extension.

```
capbakX5*option*startextension:XTESTEXT
```

To start TrueTime recording using Xtest extension.

```
capbakX5*option*startextension:XTESTEXT1EXT
```

To start TrueTime recording using Xtestext1 extension.

```
capbakX5*option*startextension:OWEXT
```

TrueTime recording using OpenWindows Extension for OpenWindows Ver 3.0

```
capbakX5*option*startextension:AUTO
```

Will select available extension for TrueTime recording (it is recommended that you choose this option if you do not want to start recording in ObjectMode).

Chapter 3 gives further information on the options in the **Output Sync Options** window, **Image Sync Options** window, **Quick Check Options** window, **Other Options** window, **Function Keys** window and **Object-Mode Options**
(See CHAPTER 3 - "CAPBAK/X Graphical User Interface” on page 37.).

CAPBAK/X Function Calls

INFORMATION: Built-in C-language functions in *CAPBAK/X*.
SEE ALSO: Appendix C, “C Interpreter Language.”
LEVEL: Intermediate to Advanced users.

B.1 Keysave File Statements

Following is a comprehensive list of keysave file statements and their definitions for *CAPBAK/X*. At the end of the chapter, sample keysave files for TrueTime, ObjectMode and “Mixed” Mode recordings are included.

Note: All statements prefixed with **`cb_`** are TrueTime function calls; statements prefixed with **`cbw_`** are ObjectMode (widget-level) function calls.

B.1.1 TrueTime Mode Function Calls

`cb_current_location_color`*(* *“white”* ***)***

Returns to 1 or 0 if current cursor location pixel color is white or not white, respectively.

`cb_delay`*(* *unsigned long int* ***)***

Waits for *n* milliseconds.

`cb_key_down`*(* *char *key, int x, int y, unsigned int n* ***)***

Identifies a key press and the length of time of that press in *n* milli-seconds.

`cb_key_str`*(* *char*key_str, char*test_name, int track_number* ***)***

Identifies a key string *key_str* from track *track_number.*

`cb_key_up`*(* *char *key, int x, int y, unsigned int delay* ***)***

Identifies a key release and the length of time of that return in *n* milliseconds.

`cb_location_color`*(* *int x, int y, “color”* ***)***

Returns to 1 or 0 if absolute location (400, 80) pixel color is white or not white, respectively.

`cb_message`*(* *char*message* ***)***

Displays *message* to output stream (defaults to *CAPBAK/X*’s message area).

`cb_error_message`*(* *char*message* ***)***

Displays *message* to *CAPBAK/X*’s message area.

`cb_bell()`

Invokes an audible bell.

`cb_mouse_button`*(char*button, int x, int y, unsigned int delay)*

Identifies a mouse button press and the length of time of that press in *n* milliseconds.

`cb_mouse_drag`*(char*button_name, char* test_name, int track_number)*

Identifies a mouse drag motion (*button_name* press) from tracking event *track_number.*

`cb_mouse_motion`*(char* test_name, int track_number)*

Identifies mouse motion from tracking event *track_number.*

`cb_mouse_move_abs`*(int x, int y, unsigned int n)*

Identifies mouse motion at *x,y* with a delay of *n* milliseconds.

`cb_save_area_ascii`*(char*test_name, int sequence number, int x, int y, int width, int height)*

This statement requires a functional OCR to capture and interpret the contents of a test area. Saves the ASCII contents of an area beginning at *x,y* for the given *width* and *height*.

`cb_save_area_image`*(char* test_name, int sequence_number, int x, int y, int width, int height)*

Saves an area that contains the point *x, y* and has a width of *width* and a height of *height* when the **Hotkey** window's **Save Mouse** button or the **F6** function key are used.

`cb_save_screen_image`(*char* test_name, int sequence_number, int x, int y, int width, int height*)

Saves the full screen that begins at *x,y* and has a width of *width* and height of *height* when the **Hotkey** window's **Save Root** button or the **F7** function key are used.

`cb_save_window_ascii`(*char*test_name, int sequence_number, int x, int y, int width, int height*)

This statement requires a functional OCR to capture and interpret the contents of a test window. Saves the ASCII contents of a window that contains the point *x,y* having the given *width* and *height*.

`cb_save_window_image`(*char* test_name, int sequence_number, int x, int y, int width, int height*)

Saves a window that contains the point *x,y* and has a width of *width* and a height of *height* when the **Hotkey** window's **Save Mouse** button or the **F6** function key are used.

`cb_sleep`(*unsigned long int*)

Sleeps for n seconds.

`cb_sync_area_ascii`(*char*string, int x, int y, width, int height*)

This statement requires a functional OCR to capture and interpret the contents of a test area. Set ASCII synchronization options in the Options--->ASCII Sync Options menu.

With ASCII Sync Options off, playback waits for an image to appear in the area that begins at *x,y* for the given *width* and *height* containing the matching string.

With ASCII Sync Options on, playback waits for an image to appear in the area that begins at *x,y* for the given *width* and *height* containing the matching string until timed out.

`cb_sync_area_image`(*char* test_name, int sequence_number, int x, int y, int width, int height*)

This statement is interpreted in one of two ways during playback:

If the **Displace Window on Image Sync** in the **Image Sync Options** window is off, waits for an image to appear in the area that begins at *x,y* has a width or *width*, a height of *height* that matches the image saved in *basename.xxx* in the `cbx_db/CBX/SYNC` sub-directory.

If the **Displace Window** on **Image Sync** in the **Image Sync Options** window is on, finds an image of the screen whose image matches that saved in *basename.xxx* (in the `SYNC` sub-directory) then moves the window so that its upper left corner is at coordinate *x,y* and has a width of *width* and a height of *height*.

`cb_sync_win_activate`(*char*window_name, int x, int y, int width, int height*)

Looks for a window named *window_name* that has just been activated and moves it so that its upper left corner is at coordinates *x*, *y* and resizes it so that is has a width of *width* and a height of *height*. This is the *CAPBAK/X* automatic output synchronization. Make sure the options are turned on in the **Output Sync Options** window in order for this statement to be interpreted during playback. Whenever a window pops up at varying locations, these options check the position of the window and relocate it if necessary. See Chapter 3 for complete information on the output synchronization options.

This statement is generated when you specify an image or window to wait to be redrawn during playback for the purposes of image synchronization. You can use the **Hotkey** window's **Sync Image** button or the **F3** function key to specify an image and the **F4** function key to specify a window.

B.1.2 ObjectMode Function Calls

`cbw_arrow_click`*(* *name, value* ***)***

Identifies an arrow click on an widget specified by *name*.

`cbw_background_color(`*"Order," "white"* **`)`**

Returns to 1 or 0 if the background color of the widget named "Order" is white or not white, respectively.

`cbw_button_click`*(* *name* ***)***

Identifies a button click by *name*.

`cbw_button_2click`*(* *name* ***)***

Identifies a button click by *name*.

`cbw_buttonG_click`*(* *name* ***)***

Identifies a button click by *name*.

`cbw_buttonG_2click`*(* *name* ***)***

Identifies a button click by *name*.

`cbw_button_set`*("* *name", "on" | "off"* ***)***

Identifies the *name* of a button and whether it is *on* or *off*.

`cbw_enter()`

Identifies the beginning of an ObjectMode recording.

`cbw_exit()`

Identifies the end of an ObjectMode recording.

`cbw_foreground_color(`*“Order”, “white”***`)`**

Returns to 1 or 0 if the foreground color of the widget named “Order” is white or not white, respectively.

`cbw_list_select(`*name, item***`)`**

Identifies when a list is selected by *name* and the *item* selected on that list.

`cbw_menu_select(`*name***`)`**

Identifies a menu is selected by *name.*

`cbw_menu_set(`*“name”, “on” | ”off”***`)`**

Identifies a menu is selected by *name*, and the action item on that menu by *on* or *off*.

`cbw_move_pointer(`*“Order”***`)`**

Moves the pointer to the widget named “Order.”

`cbw_option_select(`*name, option***`)`**

Identifes the option selection by *name*.

`cbw_scale_set(`*name, pos***`)`**

Identifies when a scale is selected by *name*, and the value *pos* it is set to.

`cbw_scrollbar_set(`*name, pos***`)`**

Identifies when a scrollbar is selected and set to value *pos*.

`cbw_sensitive(`*"Order"***`)`**

Returns to 1 or 0 if the widget named "Order" is sensitive or not sensitive, respectively.

`cbw_set_object_timeout(`*int32 i***`)`**

Identifies object timeout after *i* milliseconds.

`cbw_set_play_delay(`*int32 i***`)`**

Plays back after a delay of *i* milliseconds.

`cbw_set_retry_delay(`*int32 i***`)`**

Retry after a delay of *i* milliseconds.

`cbw_set_warp_pointer(`*int i***`)`**

Set pointer to indentifier value *i*.

`cbw_set_window(`*name***`)`**

Sets window to identifier *name*.

`cbw_set_window_timeout(`*int32 i***`)`**

Identifies windowing timeout after *i* milliseconds.

`cbw_sleep(`*delay n***`)`**

Sleeps for *n* seconds.

`cbw_submenu_select``(`*name*`)`

Identifies the selection of a submenu by *name*.

`cbw_submenu_set``(`*"name", "on" | "off"*`)`

Identifies a submenu selected by *name* and the action item on that menu by *on* or *off*.

`cbw_text_insert_pos``(`*name, row1, col1*`)`

Identifies when text is typed into any part of a widget-aware AUT, and the keystrokes entered.

`cbw_text_select_range``(`*name, row1, col1, row2, col2*`)`

Identifies the text selection by start location *row1, col1*, ending at location *row2, col2*.

`cbw_type``(`*name, str*`)`

Reads string *str* and types to window *name*.

`cbw_unpost_menu``(`*name*`)`

Deselects menu *name*.

B.1.3 Sample *CAPBAK/X* Keysave File: TrueTime Mode

```
/*
 * CAPBAK/X Release 5.2
 * (c) Copyright 1992-97, Software Research, Inc.
 * ALL RIGHTS RESERVED
 * Start of session
 */
   void test()
{
    cb_mouse_motion(“test”, 1);
    cb_mouse_button(“LEFT”, 687, 58, 349);
    cb_mouse_motion(“test”, 2);
    cb_mouse_button(“LEFT”, 358, 84, 120);
    cb_save_window_image(“test”, 2, 12, 28, 499, 316);
    cb_save_area_image(“test”, 3, 297, 594, 223, 160);
    cb_sync_area_image(“test”, 1, 153, 346, 73, 143);
    cb_mouse_button(“LEFT”, 364, 199, 491);
    cb_mouse_motion(“test”, 7);
    cb_mouse_button(“LEFT”, 622, 87, 354);
    cb_mouse_motion(“test”, 8);
    cb_mouse_button(“LEFT”, 628, 112, 252);
    cb_mouse_motion(“test”, 9);
    cb_sync_win_activate(“Options”, 547, 99, 284, 204);
    cb_mouse_motion(“test”, 10);
    cb_sync_win_activate(“Options”, 547, 99, 284, 204);
    cb_mouse_motion(“test”, 11);
    cb_mouse_button(“LEFT”, 671, 265, 156);
    cb_mouse_motion(“test”, 12);
    cb_save_screen_image(“test”, 4, 0, 0, 1152, 900);
    cb_mouse_drag(“LEFT”, “test”, 15);
    cb_mouse_drag(“LEFT”, “test”, 16);
    cb_mouse_motion(“test”, 17);
    cb_mouse_button(“LEFT”, 729, 566, 196);
    cb_mouse_motion(“test”, 18);
    cb_mouse_button(“LEFT”, 723, 545, 139);
    cb_mouse_motion(“test”, 19);
    cb_save_area_image(“test”, 5, 326, 151, 142, 106);
}
/*
 * End of session
 */
   test();    /* RUN */
```

B.1.4 Sample *CAPBAK/X* Keysave File: ObjectMode

```
/*
 * CAPBAK/X Release 5.2
 * (c) Copyright 1992-97, Software Research, Inc.
 * ALL RIGHTS RESERVED
 * Start of session
 */
   void bk()
{

 */
    cb_mouse_motion(“bk”, 5);
    cb_mouse_button(“LEFT”, 771, 502, 144);
    cb_mouse_motion(“bk”, 6);
    cbw_enter();
    cbw_set_window(“motifbur”);
    cbw_menu_select(“Edit->Select All”);
    cb_sync_win_activate(“nyi_popup”, 711, 619, 231,
113);
    cbw_set_window(“nyi_popup”);
    cbw_button_click(“Cancel”);
    cbw_set_window(“motifbur”);
    cbw_menu_select(“Order->m_show_control_button”);
    cb_sync_win_activate(“MOTIFburger Order-Entry Box”,
             517, 512, 619, 328);
    cbw_set_window(“MOTIFburger Order-Entry Box”);
    cbw_text_insert_pos(“fries_quantity”, 0, 1);
    cbw_type(“fries_quantity”, “<XK_BackSpace>11”);
    cbw_scale_set(“Quantity”, 6);
    cbw_button_set(“Mustard”, “ON”);
    cbw_button_set(“Ketchup”, “ON”);
    cbw_button_set(“Pickle”, “ON”);
    cbw_button_set(“Mayonnaise”, “ON”);
    cbw_list_select(“drink_list_box”, “Cola”);
    cbw_button_click(“up_value”);
    cbw_button_click(“up_value”);
    cbw_button_click(“up_value”);
    cbw_button_click(“Apply”);
    cbw_button_click(“Dismiss”);
    cbw_set_window(“motifbur”);
    cbw_menu_select(“Order->Submit Order”);
    cbw_menu_select(“Edit->Clear”);
    cb_sync_win_activate(“nyi_popup”, 801, 619, 231,113);
    cbw_set_window(“nyi_popup”);
    cbw_button_click(“Help”);
    cbw_button_click(“Cancel”);
    cbw_set_window(“motifbur”);
    cbw_menu_select(“File->Quit”);
    cbw_exit();
/*
 *  End of insertion
 */
```

B.1.5 Sample *CAPBAK/X* Keysave File: “Mixed” Mode

```
/*
 * CAPBAK/X Release 5.2
 * (c) Copyright 1992-97, Software Research, Inc.
 * ALL RIGHTS RESERVED
 * Start of session
 */
   void cbdemo()
{
    cb_mouse_motion(“cbdemo”, 1);
    cb_mouse_button(“LEFT”, 732, 147, 296);
    cb_save_window_image(“cbdemo”, 1, 609, 59, 499, 316);
    cb_save_area_image(“cbdemo”, 2, 636, 120, 236, 209);
    cb_mouse_button(“LEFT”, 235, 478, 415);
    cb_mouse_motion(“cbdemo”, 5);
    cb_sync_area_image(“cbdemo”, 1, 215, 475, 114, 142);
    cb_mouse_button(“LEFT”, 632, 339, 348);
    cb_save_screen_image(“cbdemo”, 3, 0, 0, 1152, 900);
    cb_key_down(“<Control_L>”, 4451);
    cb_key_up(“<Control_L>”, 659);
    cb_mouse_motion(“cbdemo”, 10);
    cb_key_down(“<Control_L>”, 2641);
    cb_key_up(“<Control_L>”, 3089);
    cb_mouse_motion(“cbdemo”, 12);
    cb_mouse_button(“LEFT”, 280, 759, 1221);
    cb_mouse_motion(“cbdemo”, 13);
}
/*
 * End of session
 */
 * Start of insertion
 */
    cb_mouse_motion(“cbdemo”, 15);
    cb_key_down(“<Control_L>”, 4614);
    cb_key_up(“<Control_L>”, 1679);
    cbw_enter();
    cbw_set_window(“motifbur”);
    cbw_menu_select(“Order”);
    cbw_menu_select(“Order->m_show_control_button”);
    cb_sync_win_activate(“MOTIFburger Order-Entry Box”);
522, 515, 619, 328);
    cbw_set_window(“MOTIFburger Order-Entry Box”);
    cbw_option_select(“Size”,”Large”);
```

```
        cbw_text_select_range(“fries_quantity”, 0, 0, 0, 1);
        cbw_type(“fries_quantity”, “<XK_BackSpace>1”);
        cbw_list_select(“drink_list_box”, “Grape Juice”);
        cbw_button_click(“up_value”);
        cbw_button_click(“up_value”);
        cbw_button_click(“Apply”);
        cbw_button_click(“Dismiss”);
        cbw_exit();
    /*
     *  End of insertion
     */
```

B.2 *CAPBAK* Utility Functions

This section describes a set of utility functions that support *CAPBAK/X* in important ways. These functions are made available as regular UNIX commands so that they can be used in *CAPBAK/X* scripts or in other products in the TestWorks family.

Inside a *CAPBAK/X* playback script, you can invoke these utility functions by including a reference to them in a system call:

`system`(*command [; command])*

Following the UNIX convention, the logical values returned by the system call (which is interpreted by the *CAPBAK/X* API) are:

0	TRUE
non-zero	FALSE

A typical passage for use inside a *CAPBAK/X* playback session could be as follows:

```
...
if (system("echo Hello world."))
{
 /* TRUE activities follow here...*/
}
{
 /* FALSE activities follow here...*/
}
...
```

B.2.1 CB.comp Utility

The **CB.comp** utility lets you compare two images to see if they are identical. (Masking is not provided; see *EXDIFF* for masking comparisons options.)

You invoke **CB.comp** as follows:

CB.comp [-help] [-q] [-Z] [-ZC *compress_cmd***]**
[-ZU *uncompress_cmd***]** *screen1 screen2*

screen1, screen2	These are the names of the files that contain the two screen images you wish to compare.
-help	Help Switch. This switch causes a detailed help screen showing sense of all arguments to be printed.
-q	Quiet Operation Switch. If present, all normal start-up output messages are suppressed.
-Z	Compression Switch. This switch turns on the image compression option. Turn on this switch if the images you want to compare are compressed image files. This switch is used if you chose automatic image compression option in recording and playing back.
-ZC *compress_cmd* **-ZU** *uncompress_cmd*	Compress/Uncompress Switch. The default compress command is **compress.** If you wish to override this default, then you specify an alternative command by naming it as *compress_cmd*. The default uncompress command is **uncompress**. If you wish to override this default, then you specify an alternative command by naming it as *uncompress_cmd*.

The resulting screen fragment from the two screen images is not displayed, as it is compared.

The return value of the **`CB.comp`** command is zero if the two images are the same. Otherwise, the return value is non-zero. This return value may be used in **Command Mode** to decide the future behavior of a playback script.

Implementation Note: `CB.comp` is a call to *EXDIFF* to obtain its results.

B.2.2 CB.find Utility

This utility finds a given screen and relocates the mouse pointer to a new location.

`CB.find` finds a given screen and gives the *x* and *y* coordinates of that screen.

You invoke **`CB.find`** as follows:

`CB.find` *screen* **`[-C | -CW | -S`** *x y***`] [-E`** *x y***`] [-help]`**
`[-m]   [-o` *outfile***`] [-q] [-t] [-Z] [-ZC`** *compress_cmd***`]`**
`[-ZU` *uncompress_cmd***`]`**

screen	Search for Screen Image Switch. This is a screen for which a match is sought. This screen image should be in standard format, such as that produced by *CAPBAK/X*.
`-C`	Start Search at Cursor Location Switch. Use the cursor and the mouse to specify upper left and lower right corners of the search area. A cross hair cursor appears and you will be prompted to mouse click on the upper left corner. Then mouse click to the right corner.

Note: **`-C`**, **`-CW`**, and **`-S`** *x y* are mutually exclusive.

`-CW`	Start Search at Cursor's Window Switch. Use the cursor to specify which window to conduct a search in. A crosshair cursor appears, and you will be prompted to mouse click on the window.

Note: **`-C`**, **`-CW`**, and **`-S`** *x y* are mutually exclusive.

`-E` *x y*	End Location Switch. This location is used as the point at which the search ends (see

below) expressed in pixels from (0, 0), the upper left hand corner of the screen.

If the values for *x* and *y* are larger than those available on the screen, then the actual lower right hand corner is used. (In effect, **-E** rounds pixel numbers down to the lower right corner of the screen.)

Note: The location specified in the **-E** switch is assumed to be below that of the start location (which could have been specified by one of the **-C**, **-CW**, and **-S** *x y* switches). If not, the search will "wrap" around the screen's lower right hand corner and may produce unexpected results.

-help Help Switch. This switch displays command summary and options on the screen.

-m Monochrome Switch. This switch forces monochrome (single bit per pixel) pattern match. Otherwise, the match is on the full image depth.

On most systems that map colors through a color tranformation, the **CB.find** algorithm for simulating monochrome comparison works in the following way:

(1) If both pixels are null, then they are considered equal.

(2) If both pixels are both non-null, then they are to be equal.

(3) If one pixel is non-null and the other is null, then they are to be different.

Two images match in the monochromatic sense even though they appear to be in different colors, as long as they both use the same all-zero null character. If you have two fragments with different colored backgrounds, they do not match.

`-o` *outfile*	Match Location Output Switch. The location of the match, if any, expressed in pixels from (0,0) at the upper left of the screen (positive numbers only), is placed in this file. If there is no match, then the *outfile* is empty. The return code of **`CB.find`** is 0 for a match, 1 for anything else.
`-q`	Quiet Operation Switch. If present, all normal start-up output messages are suppressed.
`-S` *x y*	Start Search at Specified Location Switch. Use the indicated *x*, *y* positions to start the search for a match of *screen* on the screen. The search begins at the specified locations and continues until the lower right hand corner of the screen is reached, or until the **`-E`** specified location is passed.

Note: **`-C`**, **`-CW`**, and **`-S`** *x y* are mutually exclusive.

`-t`	Tracking Indicator Switch. If present, the **`CB.find`** utility "blinks out" each pixel (and a small region around it to make the search pixel visible) to indicate where the search for a match is presently being attempted. This process slows down the search by a small amount, but gives you a visual indication that the search is continuing. At the end of the search, with or without a match, all modifications of the present

image are removed. The original image is restored 100%.

-Z	Compression Switch. This switch turns on the image compression option. Turn on this switch if the images you want to compare are compressed image files. This switch is used if you chose automatic image compression option in recording and playing back.
-ZC *compress_cmd* **-ZU** *uncompress_cmd*	Compress / Uncompress Switch. The default compress command is **`compress`**. If you wish to override this default, then you specify an alternative command by naming it *compress_cmd*. The default uncompress command is **`uncompress`**. If you wish to override this default, then you specify an alternative command by naming it *uncompress_cmd*.

B.2.3 CB.move Utility

This utility drags a particular window to a new, user-specified location.

The **`CB.move`** utility permits a *CAPBAK/X* user to "move" the top window to a specified *x*, *y* location on the screen.

You invoke **`CB.move`** as follows:

`CB.move` *x y* **`[-help] [-q]`**

x y	Location Specification. The "top window" (as determined by the window manager) is moved from its present location so that its upper left corner is positioned at *x*, *y*. The two specified values must be positive, integer numbers. The upper left hand corner of the display is *x = 0*, *y = 0*.

Note: The image being moved is adjusted so that its upper left corner, excluding the border of the image (if any), is placed in the specified *x*, *y* location. If the top window has a border, then you can take its width into account when choosing where to move it.

Warning: If the specified location is not on the screen, you *can* succeed in moving the top window *off* of the screen.

`-help`	Help Switch. This switch causes a detailed help screen showing sense of all arguments to be printed.
`-q`	Quiet Operation Switch. If present, all normal start-up output messages are suppressed.

B.2.4 CB. view Utility

The **`CB.view`** utility permits a *CAPBAK/X* user to *see* the contents of a saved screen image.

These minimal screen image processing utilities are provided for users who do not have access to the more sophisticated *EXDIFF* system for analyzing screen images.

You invoke **`CB.view`** as follows:

`CB.view` *screen* **`[-help]`** **`[-q]`** **`[-Z]`**
`[-ZC` *compress_cmd***`]`** **`[-ZU`** *uncompress_cmd***`]`**

screen	This is the name of the saved screen fragment (image) you wish to view. This screen fragment is displayed as it was originally recorded, but positioning of the image is determined by the window manager. The image persists until you type **`ctrl-c`** or click in the **`CB.view`** window.
`-help`	Help Switch. This switch causes a detailed help screen showing sense of all arguments to be printed.
`-q`	Quiet Operation Switch. If present, all normal start-up output messages are suppressed.
`-Z`	Compression Switch. This switch turns on the image compression option. Turn on this switch if the image you want to *see* is a compressed image file. This switch is used if you chose the automatic image compression option in recording and playing back.

-ZC *compress_cmd* **-ZU** *uncompress_cmd*	Compress / Uncompress Switch. The default compress command is **`compress`**. If you wish to override this default, then you specify an alternative command by naming it as *compress_cmd*. The default uncompress command is **`uncompress`**. If you wish to override this default, then you specify an alternative command by naming it as *uncompress_cmd*.

B.2.5 CB.wait Utility

This utility causes a pause until a particular image arrives at a particular location on the screen.

`CB.wait` is used to synchronize playback by waiting until a specified saved screen appears at a known location.

You invoke **`CB.wait`** as follows:

`CB.wait` *x y image timeout* **`[-help] [-q] [-Z]`**
`[-ZC` *compress_cmd*`]` **`[-ZU`** *uncompress_cmd*`]`

x	Represents the *x* coordinate in the Cartesian system.
y	Represents the *y* coordinate in the Cartesian system.
image	The file containing the *CAPBAK/X* saved image.
timeout	Timeout in seconds. If the image has not appeared in *timeout* seconds, then the system proceeds as if it *had* appeared.
`-help`	Help Switch. Displays command summary.
`-q`	Quiet Operation Switch. If present, all normal start-up output messages, such as copyright information and release information, are suppressed.

`CB.wait` will repeatedly grab from position **`<x>`**,**`<y>`** (with respect to the root window) an image of the same size as **`<image>`** and compare the grabbed image with **`<image>`** until either of the following occurs:

(1) The grabbed image is identical to **`<image>`**.

(2) The **`<timeout>`** seconds have elapsed.

	`CB.wait` waits with return value 0 if image is found within the specified time or non-zero otherwise.
`-Z`	Compression Switch. This switch turns on the image compression option. Turn on this switch if the images you want to compare are compressed image files. This switch is used if you chose automatic image compression option in recording and playing back.
`-ZC` *compress_cmd* **`-ZU`** *uncompress_cmd*	Compress/Uncompress Switch. The default compress command is **`compress`**. If you wish to override this default, then you specify an alternative command by naming it as *compress_cmd*. The default uncompress command is **`uncompress`**. If you wish to override this default, then you specify an alternative command by naming it as *uncompress_cmd*.

B.3 Standard C Library Functions

These C library functions are available with standard calling sequences for use within *CAPBAK/X* keysave files.

These functions are system-dependent; please refer to your system “man pages” to determine if they exist on your machine.

`sprintf()`

`rand ()`

`sscanf ()`

`strcat ()`

`strcpy ()`

`open ()`

`read ()`

`write ()`

`time ()`

`system ()`

`exit ()`

"C" Interpreter Language

INFORMATION: " C" interpreter language used in *SMARTS* test scripts and *CAPBAK/X* keysave files; a full description of its capabilities *and* limitations.
SEE ALSO: Appendix B, "C Functions for CAPBAK/X."
LEVEL: Intermediate to advanced users.

C.1 CINT: Purpose

SR has created a "C" interpreter language for use with the entire TestWorks product set. It works with *SMARTS* test scripts, as well as *CAPBAK*/X and *CAPBAK/MSW* keysave files. It supports "C" scalar data types, most "C" expressions, and some control-flow constructs. This interpreter will allow you to execute source code without going through the process of compiling and linking, and is helpful in the initial stages of the development process.

C.1.1 Input File Syntax

The input file format is a subset of the "C" language. The following sections describe the supported subset.

C.1.2 Data Types

This interpreter supports the following scalar types:

- **char**
- **short**
- **int**
- **long**
- **float**
- **double**
- **void**

Arrays of the scalar type are also supported.

NOTE: The following data types are ***not*** supported:

- typedefs

- structures
- unions
- enums

C.1.3 Expressions

The ANSI Standard also details the precedence and conversion rules.

C.1.4 Constants

Fixed and floating-point constants are allowed as specified by the ANSI Standard. Double-quoted strings and single-quoted characters are allowed. Long and unsigned constants are NOT supported.

C.1.5 Variables

Variables of up to 31 characters are supported with standard "C" naming conversions.

C.1.6 Operators

C.1.6.1 Supported (type double)

For the type **double**, the following "C" expression operators are supported:

- **sizeof**
- =
- +
- **-** (unary)
- **-** (binary)
- /
- |
- <
- >
- <=
- >
- = =
- **!**=
- **!**
- function call

- array reference

C.1.6.2 Supported (type int)

For the type **int**, the following "C" expression operators are supported:

- **sizeof**
- =
- +
- **-** (unary)
- **-** (binary)
- /
- **%**
- **|**
- **&** (binary)
- ^
- <
- >
- <=
- >
- = =
- **!**=
- **!**
- ++
- __
- >>
- <<
- ~
- function call
- array reference

C.1.6.3 Not Supported

The following operators are not supported:

- ?
- casts
- **-**>
- &&

- ||
- ..

C.1.7 Statements

C.1.7.1 The following statement constructs are supported:

- expressions
- **for**
- **while**
- **if**
- **break**
- **return**
- compound statements

C.1.7.2 Not Supported

The following statement constructs are not supported:

- **switch**
- **continue**
- **goto**
- **do...**
- statement labels

C.1.8 Error Messages

The interpreter supports the following diagnostic error messages. Italicized words represent parameters that are replaced by variable names or character strings.

```
1000 No user function function defined.
1001 Missing ']' in array declaration.
1002 Non integral array size expression.
1003 Expected symbol instead of data type.
1004 Syntax error in arg list. Expect symbol, not a
data type.
1005 Bad argument syntax.
1006 Can't have nested functions.
1007 Expected '{'.
1008 Missing ';' in declaration.
1009 Missing ')' in function call.
1010 Bad operand to 'argument'.
1011 Non integral operand to 'argument'.
```

```
1012 Missing ')'.
1013 Illegal function call.
1014 Need pointer or array base for subscript expression.
1015 Non-integral subscript expression.
1016 Missing ']'.
1017 Illegal Left Hand Side to assign op.
1028 Unexpected token in expression: 'expression'.
1029 Badly formed float constant.
1030 End of file before end of comment.
1031 Attempt to address past top of memory.
1032 Unknown type for variable.
1033 Unknown type in sizeof().
1034 Missing '('.
1035 Internal error in Cint, premature token list end.
1036 Symbol table overflow.
1037 Bad size to array declaration.
1036 Missing '}'.
1039 Bad function name 'function'.
1040 Too many args for function call.
1041 Arg type mismatch.
```

APPENDIX D

Imakefile: Sample from Motifburger

INFORMATION: Listing of **`motifbur`** Imakefile supplied in the demo directory.
SEE ALSO: Chapter 5, “Setup for ObjectMode Recording”.
LEVEL: Intermediate to Advanced users.

D.1 Imakefile for Motifburger

```
XCOMM
XCOMM (c) Copyright 1989, 1990, 1991, 1992 OPEN SOFTWARE FOUNDATION,
INC.
XCOMM ALL RIGHTS RESERVED
XCOMM
XCOMM
XCOMM Motif Release 1.2
XCOMM
XCOMM $RCSfile: Imakefile,v $ $Revision: 1.4 $ $Date: 92/03/13
15:37:51 $

INCLUDES = -I. -I$(INCLUDESRC) -I$(MINCLUDESRC)
 DEPLIBS = MrmClientDepLibs
LOCAL_LIBRARIES = MrmClientLibs

SRCS=              motifbur.c

XCOMM
XCOMM Software Research Notes:
XCOMM
XCOMM This is a sample Imakefile that shows you how to modify your
own
XCOMM Imakefile or Makefile to incorporate SR supplied library into
your
XCOMM application to enable it for Widget Mode testing.
XCOMM
XCOMM Depending on your platform of choice you can use the platform
specific
XCOMM flags specified within the “#ifdef <platform>” directives.
XCOMM Currently the following platforms are supported:
XCOMM - SunOS (4.1.x)
XCOMM - Solaris (5.x)
XCOMM - HP-UX (9.x)
XCOMM - IRIX (5.x)
```

```
XCOMM
XCOMM This Imakefile has 3 targets (less for certain platforms):
XCOMM motifbur.dynamic : A dynamically-linked executable. No code is
XCOMM added to the executable. Is CAPBAK/OO-aware
XCOMM by setting an environment variable to where
XCOMM SR-supplied library resides.
XCOMM motifbur.static : A statically-linked executable. The only
code
XCOMM included is the object module
XCOMM $(SR)/lib/capbakx/capbak.o. This module will
XCOMM load the necessary library at runtime.
XCOMM motifbur.complete : A complete executable. All the necessary
code
XCOMM for Widget Mode testing is included in the
XCOMM form of a library linked with the application.
XCOMM The library $(SR)/lib/capbakx/libSRcapbak.a
XCOMM should be listed before the -lXt library in
XCOMM link sequence.
XCOMM
XCOMM If you’re on a platform with no shared library support you can
only build
XCOMM the ‘motifbur.complete’ executable.
XCOMM

#ifdef HPArchitecture
STD_INCLUDES = -I/usr/include/X11R5 -I/usr/include/Motif1.2
LOCAL_LDFLAGS = -L/usr/lib/X11R5 -L/usr/lib/Motif1.2
DYNAMIC_LOCAL_LIBRARIES = -Wl,+s MrmClientLibs
STATIC_LOCAL_LIBRARIES = -Wl,-a,archive MrmClientLibs -Wl,-
a,shared,-E -ldld
COMPLETE_LOCAL_LIBRARIES = -Wl,-a,archive -L$(SR)/lib/capbakx -lSR-
capbak MrmClientLibs
#else
#if (defined(SparcArchitecture) || defined(i386Architecture)) &&
SystemV4 /* Solaris 2.4 */
STD_INCLUDES = -I/usr/openwin/include -I/usr/dt/include
LOCAL_LDFLAGS = -L/usr/openwin/lib -L/usr/dt/lib
DYNAMIC_LOCAL_LIBRARIES = MrmClientLibs
STATIC_LOCAL_LIBRARIES = MrmClientLibs -ldl
COMPLETE_LOCAL_LIBRARIES = $(SR)/lib/capbakx/libSRcapbak.a MrmCli-
entLibs
#else
#ifdef SparcArchitecture          /* SunOS 4.1.x */
STD_INCLUDES = -I/usr/X11R5/include
LOCAL_LDFLAGS = -L/usr/X11R5/lib
DYNAMIC_LOCAL_LIBRARIES = MrmClientLibs
STATIC_LOCAL_LIBRARIES = -Bstatic MrmClientLibs -Bdynamic -ldl
COMPLETE_LOCAL_LIBRARIES = -Bstatic -L$(SR)/lib/capbakx -lSRcapbak
MrmClientLibs
#else
```

```
#ifdef SGIArchitecture /* IRIX 5.x */
STD_INCLUDES = -I/usr/include
LOCAL_LDFLAGS = -L/usr/lib
DYNAMIC_LOCAL_LIBRARIES = MrmClientLibs
EXTRA_LIBRARIES = /* blank, else error when linking for next 2 libs
*/
STATIC_LOCAL_LIBRARIES = -Wl,-Bstatic $(MRESOURCELIB) $(XMLIB)
$(XTOOLLIB) -Wl,-Bdynamic -L/usr/lib $(XLIB) -lPW -ldl
COMPLETE_LOCAL_LIBRARIES = -Wl,-Bstatic -L$(SR)/lib/capbakx -lSR-
capbak $(MRESOURCELIB) $(XMLIB) $(XTOOLLIB) -Wl,-Bdynamic -L/usr/lib
$(XLIB) -lPW
#else                                                   /* Other */
STD_INCLUDES = -I/usr/include
LOCAL_LDFLAGS = -L/usr/lib
DYNAMIC_LOCAL_LIBRARIES = MrmClientLibs
STATIC_LOCAL_LIBRARIES = -Bstatic MrmClientLibs -Bdynamic -ldl
COMPLETE_LOCAL_LIBRARIES = -Bstatic -L$(SR)/lib/capbakx -lSRcapbak
MrmClientLibs
#endif
#endif
#endif
#endif


OBJS1=motifbur.o
OBJS2=motifbur.o $(SR)/lib/capbakx/capbak.o
OBJS3=motifbur.o

AllTarget(motifbur.dynamic motifbur.static motifbur.complete)

MSimpleUidTarget(motifbur)
NormalProgramTarget(motifbur.dynamic,$(OBJS1),NullParame-
ter,$(DYNAMIC_LOCAL_LIBRARIES),NullParameter)
NormalProgramTarget(motifbur.static,$(OBJS2),NullParame-
ter,$(STATIC_LOCAL_LIBRARIES),NullParameter)
NormalProgramTarget(motifbur.complete,$(OBJS3),NullParame-
ter,$(COMPLETE_LOCAL_LIBRARIES),NullParameter)

DependTarget()
```

APPENDIX E

ObjectMode Restrictions

INFORMATION: Current restrictions for ObjectMode recording.
SEE ALSO: Chapter 2, “Quick Start”.
LEVEL: All users.

E.1 Recording Limitations

ObjectMode capture has certain limitations, among them:

1. It works only on Xt-Intrinsics-based toolkits (i.e. Motif, Athena widget sets).
2. It currently works on only one application at a time.
3. It does not currently handle drawing or painting applications (for these applications, you must use the TrueTime mode of recording).
4. It currently only supports Motif 1.1 and 1.2 widget sets.
5. It does not support the use of accelerators.
6. It does not support double-clicking (to select something from a pull-down menu, you must use the “click and drag” method). This is true even in the case of “pop-up” and “pull-down” menus, where the interval between clicks is longer than usual.
7. Limited support for keyboard traversals (e.g. “up” “down” “left” “right” cursor-movement keys).
8. The HP system may not recognize the F9 key as the “End” (Stop) key during record / playback. If you are having difficulties with this setting while working on an HP, change the default setting in your SR file to F10.
9. At present, on the SGI platform, *CAPBAK/X* cannot capture images generated using the system’s graphic libraries.

10. If *CAPBAK/X* is not picking up the proper resource file, you may have to set it using the **xrdb** command:

    To set the application resources properly, type on the command line,

    (for OpenWindows):

    ```
    xrdb -merge $openwinhome/lib/app-defaults
    ```

    (for X11):

    ```
    xrdb -merge /usr/lib/X11/app-defaults
    ```

    This will establish the proper settings to do ObjectMode testing. After setting these defaults, you can invoke *CAPBAK/X*. Please see *Installation Instructions* for further information on proper setups for running *CAPBAK/X* and other STW tools.

11. To run *CAPBAK/X* (and other SR tools), you must have the proper SDK libraries on your system. This is standard on most systems; if you have questions, please contact your systems administrator.

Index