Running Web Scenarios in Headless Mode

This topic explains how to run web scenarios without opening a browser, and discusses special configuration steps for configuring this mode on Linux.

Sections include:

Running in Headless Mode

SOAtest can run web scenarios in "headless mode"—where the browser is not shown.

In command line mode (using soatestcli), SOAtest runs web scenarios in headless mode by default. If you do NOT want to run in headless mode from the cli, use the `-browserTestsVisible` command with `soatestcli` (described in Testing from the Command Line Interface - soatestcli) .

Testing from the Command Line Interface (soatestcli)

To do this:

1. Open the scenario's configuration panel.
2. In the **Browser Playback Options** tab, choose the **Headless** option.

By running tests in headless mode, you can work without the distraction of browser windows opening and closing.

Linux Configuration

To run web scenarios in headless mode in Linux, SOAtest creates its own hidden X display. SOAtest uses the X server Xvfb to create a virtual framebuffer that requires no display hardware. The SOAtest installation includes a copy of Xvfb for Linux (the file Xvfb_Linux). SOAtest will use a system-installed Xvfb if it exists. SOAtest searches the following paths, in order:

```
/usr/bin/Xvfb
/usr/X11R6/bin/Xvfb /usr/X/bin/Xvfb
/usr/openwin/bin/Xvfb
```

If your distribution does not provide Xvfb, SOAtest will use its own copy of Xvfb. This may require some configuration in the form of command line arguments. The following describes how to configure the Xvfb supplied by SOAtest to start with the appropriate arguments.

If SOAtest cannot start Xvfb, then it will simply run the browser in the display specified by the system environment variable $DISPLAY. In other words, if you are starting tests from the SOAtest GUI, the browser will appear in the same display as the GUI and SOAtest will output any startup error message from Xvfb to the Console view. If you are running tests from the command line and there is no available display, then SOAtest will not be able to start Firefox; any web scenarios will therefore fail.

Getting Xvfb working independently of SOAtest

To get Xvfb to work when invoked from SOAtest, first get Xvfb working independently of SOAtest.

(1) Start Xvfb on display :20.

Use a different display argument if :20 is not available.

```
$ cd /path/to/soatest/plugins/com.parasoft.xtest.libs.web_[version]/root
$ ./Xvfb_Linux :20 -ac
```

If you get any error messages, you will need to specify command line arguments for the location of various X-related files whose location will be dependent on your distribution. When trying Xvfb_Linux on Fedora Core, this was the error message:

```
Couldn't open RGB_DB '/usr/X11R6/lib/X11/rgb'
error opening security policy file /usr/X11R6/lib/X11/xserver/SecurityPolicy
Could not init font path element /usr/X11R6/lib/X11/fonts/misc/, removing from list!
Could not init font path element /usr/X11R6/lib/X11/fonts/Speedo/, removing from list!
Could not init font path element /usr/X11R6/lib/X11/fonts/Type1/, removing from list!
```

```
Could not init font path element /usr/X11R6/lib/X11/fonts/CID/, removing from list!
Could not init font path element /usr/X11R6/lib/X11/fonts/75dpi/, removing from list!
Could not init font path element /usr/X11R6/lib/X11/fonts/100dpi/, removing from list!

Fatal server error:
could not open default font 'fixed'
```

(2) Find the necessary X server files and add command line arguments to successfully start Xvfb

This is what worked on Fedora Core:

```
$ ./Xvfb_Linux -ac -sp /usr/lib/xserver/SecurityPolicy -fp /usr/share/X11/fonts/misc -co /usr/share/X11
/rgb -screen 0 1024x768x24 :20

A virtual frame buffer is now running on display :20.
```

(3) Try running a simple X application on the display you have created.

```
$ xclock -display :20 &
```

Verify that the clock is visible in display :20 by dumping an image of display :20 and viewing the image.

```
$ xwd -display :20 -root | xwud
```

You should see a clock. See the respective man pages for more information on xwd(1) and xwud(1).

(4) Run Firefox on the display you have created.

```
$ firefox --display :20
```

Use the same xwd/xwud command to verify that Firefox is running in the virtual frame buffer.

```
$ xwd -display :20 -root | xwud
```

If you can create an X display using Xvfb but Firefox fails to run on this virtual framebuffer, you may need to update various libraries external to SOAtest. For example, an outdated Cairo library (for 2D graphics) provided by the distribution has been known to cause problems on Linux. If you have problems with Cairo, make sure the virtual framebuffer uses a display depth greater than 8 bits because this will solve a common problem. If troubles persist, you may need to update the library itself.

Getting Xvfb working with SOAtest

Once you determine the arguments that you need to pass to Xvfb, put these arguments to use in SOAtest. To do so, create a shell script named Xvfb_Linux that invokes Xvfb with the necessary arguments. SOAtest will then run the shell script when trying to start Xvfb.

(1) In /path/to/soatest/plugins/com.parasoft.xtest.libs.web_[version]/root rename Xvfb_Linux to Xvfb_Linux.bin.

```
$ mv Xvfb_Linux Xvfb_Linux.bin
```

(2) Create a shell script Xvfb_Linux.sh that will run Xvfb_Linux.bin with the appropriate arguments.

This is what worked on Fedora Core:

```
-----
#!/bin/sh

# Use $@ to pass along any argument specified by SOAtest.
# The $@ will include the display on which to run the server.
# Currently this is always display :32.  You can override this
# by adding another display number as the last argument.

# There can be problems with graphics libraries if you do
# not set the display depth to greater than 8 using the
# -screen argument.

# Make sure to run Xvfb with 'exec' so that SOAtest will
# kill the correct process.

XVFB_DIR=`dirname $0`
exec ${XVFB_DIR}/Xvfb_Linux.bin \
  -ac \
  -sp /usr/lib/xserver/SecurityPolicy \   -fp /usr/share/X11/fonts/misc \
  -co /usr/share/X11/rgb \
  -screen 0 1024x768x24 \
  $@

-----
```

(3) Create a symbolic link to Xvfb_Linux.sh.

```
$ ln -s Xvfb_Linux.sh Xvfb_Linux
```

Then, when SOAtest runs Xvfb_Linux. it will run the script that passes the appropriate arguments to Xvfb_Linux.bin.

Alternatively, you could name the script Xvfb_Linux. However, using the symbolic link makes it easier to differentiate between what SOAtest installed and what you created to run the installed Xvfb.

(4) Run a web scenario in headless mode.

While SOAtest is running the scenario, use ps(1) to check if Xvfb_Linux.bin is running, and then use xwd/xwud again to verify that Firefox is running. (SOAtest will close Firefox as soon as it completes the test run.)

```
$ ps -ef | grep Xvfb
$ xwd -display :32 -root | xwud
```

If you configured Xvfb to run on a different display, use that when invoking xwd(1).