Known problems and solutions

External - system-side update/fix required, can't be fixed/resolved on the JDK side without it.

Specification change involved:

- [Spec][X11 compatibility]: Robot.mouseMove does not visually move mouse cursor
- [Spec][X11 compatibility]: XTest emulated mouse click does not bring window to front.
- [Spec][X11 compatibility]: Click on title to request focus test failures
- [Spec][X11 compatibility][Pure Wayland]: java.awt.Robot taking screenshots
- [Spec][X11 compatibility]: Popup menu dismiss in X11 compatibility mode.

Unresolved:

JDK-8280983 [External][Pure Wayland]: java.awt.Robot emulating keyboard/mouse events

Problem:	Cannot emulating keyboard/mouse events on Pure Wayland at all, in X11 compatibility mode only inside X11 server
Tags:	Pure Wayland(External), X11(can be used with limitations)
Related links:	**xdg-desktop-portal/issues/850** [X11 compatibility]: Robot.mouseMove does not visually move mouse cursor [X11 compatibility]: XTest emulated mouse click does not bring window to front.. [X11 compatibility]: Click on title to request focus test failures

Discussion:	The java.awt.Robot class provides methods to emulate input; keyPress(), keyRelease(), mousePress(), mouseRelease() Current robot implementation doesn't work for Pure Wayland port at all, but it mostly works for X11 compatibility mode. However, there are some cases when it fails for X11 - when trying to reach outside XWayland server with using XTest API: JDK-8280995 X11 compatibility: Robot.mouseMove does not visually move mouse cursor JDK-8280990 X11 compatibility: XTest emulated mouse click does not bring window to front. JDK-8280988 X11 compatibility: Click on title to request focus test failures Implementing this via libEI will also fix the above issues
Solution:	**X11:** Can be used in X11 compatibility mode with the above-mentioned limitations, otherwise we'll have to use the same options as for Pure Wayland. Covered by documentation changes in JDK-8307779 Relax the java.awt.Robot specification **Pure Wayland:** **Solution #1. Not yet usable** This will probably be implemented by libEI. However, its shipping time(even estimated) is not yet known. **Solution #2. Not yet usable** You can do most of this using the RemoteDesktop portal. Once permission is given, you can use it to completely control the keyboard and mouse. However, you can't keep the permission as you can with `restore_token` in the ScreenCast API, so it's not suitable for automated testing.
Workaround:	

JDK-8280984 [External][Pure Wayland]: Showing a splash screen

Problem:	Unable to control the position of the splash screen (e.g., display it in the center of the screen)
Tags:	Pure Wayland(external)
Related links:	wayland/wayland-protocols/-/issues/67
Discussion:	The native Wayland splash screen window usually appears somewhere in the upper left corner of the screen. This is due to Wayland not allowing windows to position themselves. This will need a Wayland protocol extension to add a role for an output-only, centered surface. it'd just need a fallback implementation for compositors that doesn't support it, e.g. a dialog Add input to this existing issue : wayland/wayland-protocols/-/issues/67 Implementation will probably be similar to this MR for Picture-In-Picture video: wayland/wayland-protocols/-/merge_requests/132
Solution:	None yet (see discussion in the GitLab issue).
Workaround:	use X11 if available

JDK-8280997 [External][X11 compatibility]: HiDPI.

Problem:	Applications running on XWayland looks blurry on HiDPI screens
Tags:	X11(external/internal?)

Related links:	xserver/-/issues/1318, GNOME/mutter/-/issues/402
Discussion:	Quote from Maxim Kartashev's e-mail: > *There's a quality-of-service problem with running via the compatibility layer as under certain circumstances native X windows look blurry. Users with small(ish) HiDPI displays tend to enable fractional scaling and with that enabled (regardless of the actual scale), XWayland pretends that the screen size is smaller and then pixel-stretches the resulting window according to the global scale. This works as a temporary solution, but people get quickly tired of looking at text that is blurry.* > > *See https://gitlab.gnome.org/GNOME/mutter/-/issues/402 and https://github.com/swaywm/sway/issues/5917 for some more details.* wakefield-dev/2023-February/000081.html
Solution:	wakefield-dev/2023-February/000081.html
Workaround:	

JDK-8280990 [Spec][X11 compatibility]: XTest emulated mouse click does not bring window to front.

Problem:	window does not come to front on emulated mouse click. Window focus transfer happens without issues.
Tags:	X11(spec change)
Related links:	[External][Pure Wayland][X11 compatibility]: java.awt.Robot emulating keyboard/mouse events
Discussion:	**windowDoesNotComeOnTop.c** `// gcc windowDoesNotComeOnTop.c -o windowDoesNotComeOnTop -lX11 -lXtst && ./windowDoesNotComeOnTop` `#include <stdio.h>` `#include <X11/Xlib.h>` `#include <X11/Xutil.h>` `#include <unistd.h>` `#include <X11/extensions/XTest.h>` `static Display* display;` `/*` ` * We have two windows, one overlapping other.` ` * Call to XTestFakeButtonEvent on lower window should bring it above the other one, but this doesn't happen on XWayland.` ` * Real user mouse click works fine.` ` */` `static void mapNewWindow(Window *win, int x, int y, int width, int height)` `{` `    unsigned long background = WhitePixel(display, DefaultScreen(display));` `    *win = XCreateSimpleWindow(display, DefaultRootWindow(display), x, y, width, height, 0, 0, background);` `    XSizeHints    hints = {0};` ` hints.flags = PPosition	PSize;` ` hints.x = x;` ` hints.y = y;` ` hints.width = width;`

```
    hints.height = height;

    XSetNormalHints(display, *win, &hints);

    XSelectInput(display, *win, ButtonPressMask | ButtonReleaseMask | StructureNotifyMask);
    XMapWindow(display, *win);
}

void clickOnLeftWindow(Display* display) {
    int x = 225;
    int y = 150;

    printf("XWarpPointer to %d %d\n", x, y);

    XWarpPointer(display, None, DefaultRootWindow(display), 0, 0, 0, 0, x, y);
    XSync(display, False);

    printf("XTestFakeButtonEvent\n");
    XTestFakeButtonEvent(display, 1, True, CurrentTime);
    XSync(display, False);

    XTestFakeButtonEvent(display, 1, False, CurrentTime);
    XSync(display, False);
}

int main(void) {
    XInitThreads();
    display = XOpenDisplay(NULL);

    int width = 200;
    int height = 100;

    Window w1, w2;
    mapNewWindow(&w1, 200, 50, width, height);
    mapNewWindow(&w2, 350, 50, width, height);

    XFlush(display);

    int mappedCount = 0;

    while (mappedCount < 2) {
        XEvent e;
        XNextEvent(display, &e);

        switch (e.type) {
            case MapNotify: {
                printf("%lx: MapNotify\n", e.xmap.window);
                mappedCount++;
                break;
            }
            default:
                printf("%lx: Received event type: %i\n",  e.xany.window,  e.type);

        }
    }

    sleep(2);
    clickOnLeftWindow(display);

    while (True) {
        XEvent e;
        XNextEvent(display, &e);

        switch (e.type) {
            case ButtonPress:
            case ButtonRelease: {
                printf("%lx: Button%s\n",
                       e.xbutton.window,
                       e.type == ButtonRelease ? "Release" : "Press"
                );
                break;
```

```
            }
            default:
                printf("%lx: type: %i\n",  e.xany.window, e.type);

        }
    }
}
```

> *Yes, the reason is this is using XTEST, an X11 protocol which will not work outside of X11.*
>
> *In other words, the emulated input event reaches the X11 clients, but not the Wayland compositor which is the actual display server but also the X11 window manager in Wayland, the component which is in charge of moving/resizing/stacking the windows.*
>
> *You can easily observe this using xdotool and xev in Xwayland. If the emulated click occurs within the xev window, the X11 event is logged by xev, but the window is not restacked by the Wayland compositor.*

Solution:	Use the same solution as for the Pure Wayland toolkit Preferred: emulate mouse click with libei
Workaround:	modify tests to use another API to bring window to front (e.g XMapRaised) Covered by documentation changes in JDK-8307779 Relax the java.awt.Robot specification. Test modifications are still needed.

JDK-8280988 [Spec][X11 compatibility]: Click on title to request focus test failures

Problem:	Some tests are trying to get focus of a window by clicking on its title bar.
Tags:	X11(spec change)
Related issues:	[External][Pure Wayland][X11 compatibility]: java.awt.Robot emulating keyboard/mouse events
Discussion:	
Solution:	Use the same solution as for the Pure Wayland toolkit
Workaround:	Click on window's body instead of decorations, request focus in other way Covered by documentation changes in JDK-8307779 Relax the java.awt.Robot specification. Test modifications are still needed.

JDK-8283278 [External][X11 compatibility]: XOR rendering issue

Problem:	window is not rendered correctly(wrong colors) using setXORMode in Virtual Box with 3D acceleration enabled.
Tags:	X11(external)
Related links:	xorg/xserver/-/issues/1333 mesa/mesa/-/issues/6596
Discussion:	Reproducers: java and native E.g. how should be displayed the native reproducer : blocked URL And how it actually looks like: blocked URL
Solution:	None, waiting for external fix
Workaround:	

JDK-8280987 [X11 compatibility]: crash when mapping a lot of windows

Problem:	GUI are crashing if you are trying to map a lot(> ~260) of windows at once.
Tags:	X11(external)
Related links:	xorg/xserver/-/issues/1222.

Discussion:	`// gcc windowSpawnDoS.c -o windowSpawnDoS -lX11 && ./windowSpawnDoS`
	`#include <X11/Xlib.h>`
	`#include <unistd.h>`
	
	`int main(void) {`
	`  XInitThreads();`
	`  Display* dpy = XOpenDisplay(NULL);`
	
	`  Window root = DefaultRootWindow(dpy);`
	`  for (int i = 0; i < 500; ++i) {`
	`  Window ww = XCreateSimpleWindow(dpy, root, 50, 50, 50, 50, 0, 0, 0);`
	`  XMapWindow(dpy, ww);`
	`  }`
	
	`  for(;;) {`
	`  XEvent e;`
	`  XNextEvent(dpy, &e);`
	`  }`
	`}`
	
	It may be faced when running some tests which are trying to map a lot of windows, e.g. one for every GraphicsConfiguration.
	
	If you have 210 GraphicsConfigurations per display on Xwayland, then running such test in 2 display configuration will lead to crash.

Solution:	None (yet)
Workaround:	Fix the failing test, e.g. by adding some delay after each ~100 windows displayed

Fix in progress:

Fixed:

JDK-8280982 [Spec][X11 compatibility][Pure Wayland]: java.awt.Robot taking screenshots

Problem:	The java.awt.Robot class provides createScreenCapture() method which currently uses X11-specific APIs
Tags:	X11, Pure Wayland, spec change
Related links:	
Discussion:	It is doable as of now, but not in a standard and convenient way. It can be done using some shell specific APIs, e.g. org.gnome.Shell.Screenshot DBUS API (which will not work for KDE, it has another API) It has several drawbacks: • There is no direct access to the screenshot data. Screenshot is saved to `png` file, after that you have to read it, decode it. • Filesystem footprint. You need to save it to some file(its filesystem can be slow). It can be workarounded by using shared memory object though. • Using DBUS API in this case is really slow. For instance, just a sync DBUS call to take a 1000x1000 screenshot and save it to the shared memory object file is ~14 time slower(it is just saving, without decoding and extracting data) comparing to full X11 implementation. So, it would be great to get some standard API to get screenshot data in memory. I see that there is an open bug against this issue. Specification change needed to handle user confirmation of screen capture (user can deny/partially allow screen capture).

Solution:	**Solution #1.**
	
	Specifically, it provides org.freedesktop.portal.Screenshot#Screenshot which can be used to create a screenshot.
	
	However, it has some drawbacks(at least on Gnome):
	
	• It does not allow to take a specified screen area, only whole screen.
	• ~~It asks a confirmation on each screenshot request, so it is unusable for automated testing.~~ (note, this was fixed for GNOME 43, not yet delivered to most distros)
	• It saves a screenshot to disk, which must be read and deleted(may be slow depending on drive).
	• A user can deny the request. In this case as a possible solution we can throw a SecurityException for Robot#createScreenCapture (javadoc update required). Currently it is thrown only if readDisplayPixels permission is not granted(which is not suitable here).
	• There is no way to disable the "screen flash" after screenshot
	
	There is also org.freedesktop.portal.Screenshot#PickColor API which can't be used for Robot#getPixelColor needs. It does not allow to take a pixel color at specified location, but interactively asks a user to pick a color with mouse cursor.
	
	Solution #2. Preferred
	
	Finally, another possibility is to use the Screencast portal.
	
	This might be a bit of overkill, but it avoids several of the problems mentioned in the screenshot and color picker portals.
	
	• implementation is more complicated comparing to Screenshot API
	• A user can deny the screenshot request too.
	  ◦ Permission to make screenshots can be stored with restore_token (string, introduced in xdg-desktop-portal 1.12), that needs to be stored safely somewhere.
	  ◦ new permission required if display set is changed or rearranged, might be an issue in case of remote automated testing.
	• no intermediate file, screenshot data can be obtained from memory
	• each display has its own stream, in case if requested screenshot area covers several displays resulting screenshot need to be combined from pieces.
	
	Each solution is viable, but #2 seems to be more preferable.
	
	However we may want to implement both of approaches:
	
	If some new display got or disconnected, new permission request from user is required. In case of automated testing it may be a stopper. But we can fallback to a solution #1.
	
	Prototype is available here.
	
	• `-Dawt.robot.screenshotMethod=dbusScreencast` to switch on screenshot via Screencast(x11 for switch off). Enabled by default on Wayland
	• `-Dawt.robot.screenshotDebug=true` to print debug info, disabled by default
	
	CSR Approved JDK-8307456
	
	Review in process: openjdk/jdk/pull/13803

Workaround:	

Testing on **latest supported Linux distros for JDK19**

Packages required for permission restoration:

- xdg-desktop-portal 1.12+
- xdg-desktop-portal-gnome 42+ (it is a backend for xdg-desktop-portal)

However xdg-desktop-portal-gnome package is installed only for Ubuntu and SLES, OEL9 and RHEL9 does not have it yet.

	works?	xdg-desktop-portal	xdg-desktop-portal-gnome	Gnome	
Ubuntu 22.04	Yes	1.14.4-1	42.1	42.2	
OEL9	Partially	1.12.1-1	missing	40.4.0	persist_mode and restore_token has no effect
RHEL9	Partially	1.12.1-1	missing	40.4.0	persist_mode and restore_token has no effect
SLES15	Partially	1.10.1	41.1	41.2	protocol version 3 (1.10 < 1.12), restore_token is not supported

Where partially means that you can make a screenshot, however it requires user permission on each request.

JDK-8280994 [External][X11 compatibility]: Drag and Drop java Wayland does not work

Problem:	Drag and drop from java to Wayland apps does not work with current JDK implementation.
Tags:	X11(external)
Related links:	GNOME/mutter/-/issues/2042, GNOME/mutter/-/merge_requests/2124
Discussion:	JDK code it quite picky for the drop target, and requires it to be a toplevel with WM_STATE property, whereas the Wayland server uses dummy windows without this property set. We need to disable this behaviour when the Wayland session is active.
Solution:	external fix released, fix on JDK side targeted for 21
Workaround:	

JDK-8280995 [Spec][X11 compatibility]: Robot.mouseMove does not visually move mouse cursor

Problem:	Robot.mouseMove does not visually move mouse cursor
Tags:	X11(spec change)
Related links:	[External][Pure Wayland][X11 compatibility]: java.awt.Robot emulating keyboard/mouse events

Discussion:	Javadoc for `java.awt.Robot#mouseMove` says:
	
	`/**`
	` * Moves mouse pointer to given screen coordinates.`
	` * @param x X position`
	` * @param y Y position`
	` */`
	`public synchronized void mouseMove(int x, int y) {`
	
	Despite the fact that this mouse movement is successfully emulated within the XWayland server(following mouse press emulation will be in a right place), the mouse cursor does not visually move.
	
	With current specification of `java.awt.Robot#mouseMove` it may be a conformance issue.

Solution:	Use the same solution as for the Pure Wayland toolkit Preferred: see earlier point on working with RemoteDesktop portal(can't keep the user permission between sessions) JDK21: Covered by JDK-8307779 Relax the java.awt.Robot specification
Workaround:	Change the documentation to reflect the current behavior

JDK-8280993 [Spec][X11 compatibility]: Popup menu dismiss in X11 compatibility mode.

Problem:	We are using `XGrabPointer` to get mouse input and dismiss popup menus on mouse click. Obviously, it does not work outside of XWayland server.
Tags:	X11, spec change
Related links:	

Discussion:	E.g. if we have shown some popup menu, it will be closed upon clicking on any of XWayland's windows.
	
	But it will not be closed if we click on some other window from Wayland.
	
	`diff --git a/src/java.desktop/unix/classes/sun/awt/X11/XPopupMenuPeer.java b/src/java.desktop/unix/classes/sun/awt/X11`
	`/XPopupMenuPeer.java`
	`index a19f56249ae..db88ef49f37 100644`
	`--- a/src/java.desktop/unix/classes/sun/awt/X11/XPopupMenuPeer.java`
	`+++ b/src/java.desktop/unix/classes/sun/awt/X11/XPopupMenuPeer.java`
	`@@ -111,6 +111,16 @@ public class XPopupMenuPeer extends XMenuWindow implements PopupMenuPeer {`
	`         // Get menus from the target.`
	`         Vector<MenuItem> targetItemVector = getMenuTargetItems();`
	`         if (targetItemVector != null) {`
	`+            //TODO: add focus listener only for XWayland`
	`+            target.addFocusListener(new FocusAdapter() {`
	`+                @Override`
	`+                public void focusLost(FocusEvent e) {`
	`+                    target.removeFocusListener(this);`
	`+                    if (isShowing()) {`
	`+                        hide();`
	`+                    }`
	`+                }`
	`+            });`
	`             reloadItems(targetItemVector);`
	`             //Fix for 6287092: JCK15a: api/java_awt/interactive/event/EventTests.html#EventTest0015 fails, mustang`
	
	For a first look this workaround seems to work reliably except only one case:
	
	clicking on window's title containing origin component does not lead to focus change, thus we don't hiding a popup.
	
	I see a behavior similar to this workaround in some other third party application running on XWayland.

Solution:	Use workaround. This also requires a change in documentation, as there are "blind spots" which, if clicked on, will not close the popup window(e.g. window title or area in the side dock without application icons) Fixed in JDK21
Workaround:	Dismiss menu on window focus lost event(Seems to be a common workaround, third party X11 applications already use it))

JDK-8280985 [External][X11 compatibility]: Wayland crash on huge window

Problem:	System session crashes when trying to display a huge window
Tags:	X11(external)
Related links:	xorg/xserver/-/merge_requests/754, wayland/wayland/-/merge_requests/179

Discussion:	`// gcc hugeFrame.c -o hugeFrame -lX11 && ./hugeFrame`
	`#include <X11/Xlib.h>`
	`#include <unistd.h>`
	
	`static Display* display;`
	
	`int main(void) {`
	`  int width = 22000;`
	`  int height = 25000;`
	
	`  display = XOpenDisplay(NULL);`
	
	`  Window win = XCreateSimpleWindow(display, DefaultRootWindow(displ ay), 0, 0, width, height, 0, 0L, 0L);`
	`  XMapWindow(display, win);`
	`  XFlush(display);`
	
	`  sleep(1);`
	`}`
	
	`/var/log/syslog` shows:
	
	`gnome-shell[1248]: WL: error in client communication (pid 1248)`
	`gnome-shell[1298]: (EE)`
	`gnome-shell[1298]: Fatal server error:`
	`gnome-shell[1298]: (EE) wl_shm@5: error 1: invalid size (-2094967296)`
	`gnome-shell[1298]: (EE)`
	
	where -2094967296 looks like an integer overflow of 4 * 22000 * 25000
	
	So it fails to create a pool, but doesn't handle it gracefully.

Solution:	external fix released
Workaround:	

JDK-8281612 [External][X11 compatibility]: Drag and Drop, drop performs to a wrong window from X11 client

Problem:	Can't perform a Drag and Drop operation from X11 client to a Wayland client if there is an intermediate window during the drag.
Affected:	X11(external)
Related links:	GNOME/mutter/-/issues/2136
Discussion:	
Solution:	external fix released
Workaround:	

JDK-8280992 [External][X11 compatibility]: Custom cursor might be displayed in wrong color

Problem:	Custom cursor(e.g. loaded from GIF) set by Component#setCursor might be displayed in wrong color.
Tags:	X11(external)
Related links:	/xorg/xserver/-/issues/1303, MR
Discussion:	For example on following screenshot X shaped cursor should be displayed in yellow: blocked URL Reported as /xorg/xserver/-/issues/1303
Solution:	external fix released
Workaround:	

JDK-8280991 [External][X11 compatibility]: fake configure event for XRandR emulation

Problem:	We are not getting updates from the system after "changing" screen resolution via `XRRSetScreenConfigAndRate`.
Tags:	X11 (external)
Related links:	xorg/xserver/-/merge_requests/731, xorg/xserver/-/issues/1305
Discussion:	Excerpt from Olivier's mail: > *Also worth noting that the XRandR emulation is per window/X11 client, whereas the root window is shared between all X11 clients, but maybe we could send a fake ConfigureNotify event to the given client, I would need to check if that's doable.* This notification would be helpful. This MR adds notification, however it has several issues: 1. There is no event when you are trying to change to native resolution from another. 2. `XRRScreenChangeNotifyEvent` always has native resolution reported. (we are not using it though) 3. It might be a minor one, but on X11 session there is no events reported if you are trying to change to the same resolution.
Solution:	external fix released
Workaround:	