Mir

Release v2.16.4

Canonical Group Ltd.

Apr 04, 2024

CONTENTS

Mir is a next generation display server targeted as a replacement for the X window server system to unlock next-generation user experiences for devices ranging from Linux desktop to mobile and IoT devices powered by Ubuntu.

- If you want to use the Mir snaps, see: Make a Secure Ubuntu Web Kiosk
- If you want to try out the Mir demos on desktop, see: *Getting and using Mir*
- If you want to get involved in Mir development, see: *Getting involved in Mir*

CHAPTER

ONE

USING MIR FOR SERVER DEVELOPMENT

Install the headers and libraries for using libmiral in development:

```
sudo apt install libmiral-dev
```

A `miral.pc` file is provided for use with `pkg-config` or other tools. For example:

```
pkg-config --cflags miral
```

The server API is introduced here: Introducing the MirAL API

1.1 Getting and Using Mir

Mir is a library for building things, not an end-user product, but it does come with some demos to illustrate the possibilities.

1.1.1 Getting Mir demos

The Mir libraries and demos are available on Ubuntu, Fedora and Arch. It has also been built and tested on Debian but, at the time or writing, is not in the archive.

For Linux distributions that don't currently package Mir you need to build it yourself. (See *Getting Involved in Mir*).

1.1.2 Getting Mir on Ubuntu

You can install the Mir examples along with the Mir graphics drivers as follows:

```
sudo apt install mir-demos mir-graphics-drivers-desktop
```

It is also useful to install Wayland support for Qt and qterminal:

```
sudo apt install qterminal qtwayland5
```

Getting the latest Mir release on Ubuntu

As a matter of policy Ubuntu does not routinely update packages after a series is released. This means the Mir team cannot reasonably guarantee that all series will have the latest Mir release available at all times.

The latest Mir release is available for all supported Ubuntu series from the Mir team's "release PPA". To add the PPA to your system:

```
sudo add-apt-repository --update ppa:mir-team/release
```

To remove the PPA from your system:

```
sudo ppa-purge mir-team/release
```

1.1.3 Getting Mir on Fedora

On Fedora Mir is available from the archive:

```
sudo dnf install mir-demos
```

It is also useful to install qterminal:

```
sudo dnf install qterminal
```

1.1.4 Getting Mir on Arch

On Arch Linux, you can install the mir package from the AUR.

1.1.5 Using Mir demos

For convenient testing under X11 there's a "miral-app" script that wraps the commands used to start a server and then launches a terminal (as the current user):

```
miral-app
```

If you're using a desktop that supports X11 then you can run this in a terminal window. In that case Mir will automatically select a "Mir-on-X11" backend and run in a Window.

Alternatively, to run Mir "natively" you can run the same command in a Virtual Terminal.

Running applications on Mir

If you use the terminal launched by `miral-app` Wayland applications can be started as usual:

```
kate
neverputt
gedit
```

Options when running the Mir example shell

Script Options

The `miral-app` script provides options for using an alternative shell (e.g. `miral-kiosk` as used by the mir-kiosk snap) and an alternative terminal.

```
-kiosk                      use miral-kiosk instead of miral-shell
-terminal <terminal>        use <terminal> instead of '/usr/bin/miral-terminal'
```

The default for `-terminal` is a script that tries to identify the system terminal emulator and launch that. But another terminal, or indeed any application, can be used.

For example:

```
miral-app -kiosk -terminal supertuxkart
```

There are some additional options (listed with "-h") but those are the important ones.

`miral-shell` Options

The script will pass everything on the command-line following the first thing it doesn't understand to `miral-shell`. The options can be listed by `miral-shell --help`. The following are likely to be of interest:

```
--window-management-trace           log trace message
```

Probably the main use for `miral-shell` is to test window-management (either of a client toolkit or of a server) and this logs all calls to and from the window management policy. This option is supported directly in the MirAL library and works for any MirAL based shell - even one you write yourself.

```
--window-manager arg (=floating)   window management strategy
                                   [{floating|tiling|system-compositor}]
```

This allows an alternative "tiling" window manager to be selected. *Note: `--window-manager` is only supported by `miral-shell` (not `miral-kiosk`).*

For example:

```
miral-app --window-manager tiling
```

These options can also be specified in a configuration file. For example:

```
$ cat ~/.config/miral-shell.config
keymap=gb
window-manager=tiling
```

1.2 Getting Involved in Mir

1.2.1 Getting involved

The Mir project website is https://mir-server.io/, the code is hosted on GitHub

For announcements and other discussions on Mir see: Mir on community.ubuntu

For other questions and discussion about the Mir project: the #mir-server IRC channel on Libera.Chat.

1.2.2 Getting Mir source and dependencies

You can get the source with:

```
git clone https://github.com/MirServer/mir.git
cd mir
```

You may need to install git for the system you are working on.

You'll also need a few development tools installed. The exact dependencies and instructions vary across distros.

On Ubuntu

```
sudo apt-get build-dep ./
```

On Fedora and Alpine

As we build these distros in Mir's CI you can copy the instructions from the corresponding files under `spread/build`.

1.2.3 Building

```
cmake -S . -Bbuild
cd build
cmake --build .
```

This creates an example shell (miral-shell) in the bin directory. This can be run directly:

```
bin/miral-app
```

With the default options this runs in a window on X (which is convenient for development).

The miral-shell example is simple, don't expect to see a sophisticated launcher by default. Within this window you can start a terminal with Ctrl-Alt-Shift-T (the "Shift" is needed to avoid Ubuntu's DE intercepting the input). From this terminal you can start apps. For example:

```
$ gedit
```

To exit from miral-shell press Ctrl-Alt-BkSp.

You can install the Mir examples, headers and libraries you've built with:

```
sudo cmake --build . --target install
```

1.2.4 Contributing to Mir

Please file bug reports at: https://github.com/MirServer/mir/issues.

The Mir Discourse category can be found at: https://discourse.ubuntu.com/c/mir/15.

1.2.5 Working on code

- Hacking guidelines can be found here: Mir Hacking Guides
- You can configure *Mir* to provide runtime information helpful for debugging by enabling component reports: Component Reports
- A guide on versioning Mir DSOs: DSO Versioning Guide

1.3 Architecture

This document introduces the architecture of *Mir* at a high-level.

1.3.1 Audience

This document is intended to provide contributors to *Mir* an overview of *Mir*'s systems. It is *not* intended to guide compositor authors.

1.3.2 Index

1.3.3 APIs for compositor authors

Mir provides compositor authors with a set of libraries that they can use to build Wayland based shells. These libraries are:

- *miral*
- *miroil*
- *mirwayland*

Miral

The most commonly used library is **miral**. `miral` (the "*Mir* Abstraction Layer") is an API that makes *Mir* easy for compositor authors to work with. It provides core window management concepts and an interface that is more ABI stable than Mir's internal API. While `miral` is built on the Mir *engine*, compositor authors are encouraged to only interact with the high-level `miral` library.

Miroil

miroil is a custom library for the Lomiri folks. It is like `miral` in that it provides an abstraction over the Mir *engine*. However, most compositor authors will not interact with `miroil`.

Mirwayland

Compositor authors may want to define their own wayland protocol extensions in addition to the ones that the core *Mir* implementation defines. The `mirwayland` library satisfies this requirement. This library may be used in conjunction with either `miral` or `miroil`.

1.3.4 The Mir engine

The **mirserver** library is the core implementation of *Mir*. It serves as the engine for both `miral` and `miroil`. This library does the heavy-lifting for the compositor and, as a result, is the largest piece of *Mir*. This section will explain the primary concepts that drive the engine.

Core Concepts

At the heart of `mirserver` are two interfaces: the **Shell** and the **Scene**. The `Shell` is responsible for fielding requests from the rest system. The `Shell` modifies the state of the `Scene` to reflect the requested changes.

For example, the `Frontend` would ask the `Shell` to initiate dragging a window. The `Shell` would then decide how to move that window to and update the state of the `Scene` to reflect that change.

From Scene to Screen

Knowing that the `Scene` holds the state of what is to be displayed, we can talk about the **Compositor**. The `Compositor` gets the collection of items to render from the `Scene`, renders them with the help of the *rendering platform*, and sends them off to the *display platform* to be displayed.

From Interaction to Shell

As stated previously, the `Shell` handles requests from the system and updates the state of the `Scene` accordingly. These requests come from a variety of sources, which we will investigate now.

Frontend Wayland: Responsible for connecting the Wayland protocol with the rest of *Mir*. The `WaylandConnector` class connects requests made via the Wayland protocol to the core state of the compositor.

Frontend XWayland: Responsible for connecting requests sent by an `XWayland` server to the rest of *Mir*. The `XWaylandConnector` establishes this connection. This frontend spawns an `XWayland` server as a subprocess.

Input: Handles everything having to do with input, including mouse, keyboard, touch, and more. This system interacts with the specific *input platform* and bubbles up events through a list of `InputDispatcher`s, which enables different pieces of the software to react to events.

For example, a compositor's window manager may respond to a key press event by opening up a new terminal via a request to the `Shell`.

1.3.5 Platforms

We briefly hinted at the existence of so-called "platforms" previously, but they are deserving of a dedicated section. A **Platform** is an adapter that allows the system to work across different graphics, input, and rendering stacks. They come in three flavors:

- **Display Platform**: Determines what the compositor is rendering to. This may be a physical monitor via GBM/KMS (or EGLStreams for Nvidia), an X11 or Wayland window, or a completely virtual buffer.
- **Input Platform**: Determines where the compositor is getting input from. This could be native event via `libinput`, X input events, or Wayland input events.
- **Rendering Platform**: Determines how the compositor renders the final image. For now, only a GL backend is supported.

The GBM/KMS platform is most typically what will be used, as it is the native platform. The X11 platform is useful for development. The Wayland platform is specifically useful for Ubuntu Touch, where they are hosting *Mir* in another Wayland compositor.

1.3.6 Supporting Libraries

Mir leans on a few core libraries to support the entire system. These libraries contain data structures and utilities that are shared throughout the project, including `miral` and `miroil`.

- **Core**: Fundamental data concepts, like file descriptors and rectangles. These data structures tend not to be *Mir*-specific.
- **Common**: *Mir*-specific data concepts like *Mir* event building, logging, and timing utilities.

1.4 Mir API

1.4.1 Page Hierarchy

- page_deprecated

1.4.2 Class Hierarchy

- *Namespace mir*
  - *Namespace mir::geometry*
    - *Namespace mir::geometry::generic*
      - *Template Struct Displacement*
      - *Template Struct Point*
      - *Template Struct Rectangle*
      - *Template Struct Size*
      - *Template Struct Value*

- *Enum MirTouchpadScrollMode*
- *Enum MirTouchscreenMappingMode*
- *Enum MirTouchTooltype*
- *Enum MirWindowAttrib*
- *Enum MirWindowFocusState*
- *Enum MirWindowState*
- *Enum MirWindowType*
- *Enum MirWindowVisibility*

1.4.3 File Hierarchy

- **dir_examples**
  - **dir_examples_example-server-lib**
    - file_examples_example-server-lib_decoration_provider.h
    - file_examples_example-server-lib_floating_window_manager.h
    - file_examples_example-server-lib_splash_session.h
    - file_examples_example-server-lib_sw_splash.h
    - file_examples_example-server-lib_tiling_window_manager.h
    - file_examples_example-server-lib_wallpaper_config.h
    - file_examples_example-server-lib_wayland_app.h
    - file_examples_example-server-lib_wayland_shm.h
    - file_examples_example-server-lib_wayland_surface.h
  - **dir_examples_miral-kiosk**
    - file_examples_miral-kiosk_kiosk_window_manager.h
  - **dir_examples_miral-shell**
    - **dir_examples_miral-shell_spinner**
      - file_examples_miral-shell_spinner_eglapp.h
      - file_examples_miral-shell_spinner_miregl.h
      - file_examples_miral-shell_spinner_splash.h
- **dir_include**
  - **dir_include_core**
    - **dir_include_core_mir**
      - **dir_include_core_mir_geometry**

        file_include_core_mir_geometry_dimensions.h

        file_include_core_mir_geometry_displacement.h

        file_include_core_mir_geometry_forward.h

        file_include_core_mir_geometry_point.h

file_include_core_mir_geometry_rectangle.h

file_include_core_mir_geometry_rectangles.h

file_include_core_mir_geometry_size.h

- file_include_core_mir_abnormal_exit.h
- file_include_core_mir_anonymous_shm_file.h
- file_include_core_mir_depth_layer.h
- file_include_core_mir_fatal.h
- file_include_core_mir_fd.h
- file_include_core_mir_int_wrapper.h
- file_include_core_mir_optional_value.h
- file_include_core_mir_proof_of_mutex_lock.h
- file_include_core_mir_shm_file.h
- file_include_core_mir_synchronised.h

* **dir_include_core_mir_toolkit**

  - **dir_include_core_mir_toolkit_events**

    file_include_core_mir_toolkit_events_enums.h

  - file_include_core_mir_toolkit_common.h
  - file_include_core_mir_toolkit_mir_input_device_types.h
  - file_include_core_mir_toolkit_mir_native_buffer.h
  - file_include_core_mir_toolkit_mir_version_number.h

– **dir_include_miral**

* **dir_include_miral_miral**

  - file_include_miral_miral_add_init_callback.h
  - file_include_miral_miral_append_event_filter.h
  - file_include_miral_miral_application.h
  - file_include_miral_miral_application_authorizer.h
  - file_include_miral_miral_application_info.h
  - file_include_miral_miral_canonical_window_manager.h
  - file_include_miral_miral_command_line_option.h
  - file_include_miral_miral_configuration_option.h
  - file_include_miral_miral_cursor_theme.h
  - file_include_miral_miral_display_configuration.h
  - file_include_miral_miral_display_configuration_option.h
  - file_include_miral_miral_external_client.h
  - file_include_miral_miral_internal_client.h
  - file_include_miral_miral_keymap.h

- file_include_miral_miral_lambda_as_function.h
- file_include_miral_miral_minimal_window_manager.h
- file_include_miral_miral_output.h
- file_include_miral_miral_prepend_event_filter.h
- file_include_miral_miral_runner.h
- file_include_miral_miral_set_command_line_handler.h
- file_include_miral_miral_set_terminator.h
- file_include_miral_miral_set_window_management_policy.h
- file_include_miral_miral_toolkit_event.h
- file_include_miral_miral_version.h
- file_include_miral_miral_wayland_extensions.h
- file_include_miral_miral_window.h
- file_include_miral_miral_window_info.h
- file_include_miral_miral_window_management_options.h
- file_include_miral_miral_window_management_policy.h
- file_include_miral_miral_window_manager_tools.h
- file_include_miral_miral_window_specification.h
- file_include_miral_miral_x11_support.h
- file_include_miral_miral_zone.h

– dir_include_miroil

∗ dir_include_miroil_miroil

- file_include_miroil_miroil_compositor.h
- file_include_miroil_miroil_display_configuration_controller_wrapper.h
- file_include_miroil_miroil_display_configuration_policy.h
- file_include_miroil_miroil_display_configuration_storage.h
- file_include_miroil_miroil_display_id.h
- file_include_miroil_miroil_display_listener_wrapper.h
- file_include_miroil_miroil_edid.h
- file_include_miroil_miroil_event_builder.h
- file_include_miroil_miroil_eventdispatch.h
- file_include_miroil_miroil_input_device.h
- file_include_miroil_miroil_input_device_observer.h
- file_include_miroil_miroil_mir_prompt_session.h
- file_include_miroil_miroil_mir_server_hooks.h
- file_include_miroil_miroil_mirbuffer.h
- file_include_miroil_miroil_open_gl_context.h

- file_include_miroil_miroil_persist_display_config.h
- file_include_miroil_miroil_prompt_session_listener.h
- file_include_miroil_miroil_prompt_session_manager.h
- file_include_miroil_miroil_set_compositor.h
- file_include_miroil_miroil_surface.h
- file_include_miroil_miroil_surface_observer.h

1.4.4 Full API

Namespaces

Namespace mir

Namespaces

- *Namespace mir::geometry*

Classes

- *Struct IntOwnedFd*
- *Class AbnormalExit*
- *Class AnonymousShmFile*
- *Class ExitWithOutput*
- *Class FatalErrorStrategy*
- *Class Fd*
- *Template Class IntWrapper*
- *Template Class optional_value*
- *Class ProofOfMutexLock*
- *Class ShmFile*
- *Template Class Synchronised*
- *Template Class Synchronised::LockedImpl*

Functions

- *Function mir::fatal_error_abort*
- *Function mir::fatal_error_except*
- *Function mir::mir_depth_layer_get_index*
- *Template Function mir::operator!=(T const&, optional_value<T> const&)*
- *Template Function mir::operator!=(optional_value<T> const&, T const&)*

- *Template Function mir::operator!=(IntWrapper<Tag, ValueType> const&, IntWrapper<Tag, ValueType> const&)*
- *Template Function mir::operator!=(optional_value<T> const&, optional_value<T> const&)*
- *Template Function mir::operator<*
- *Template Function mir::operator<<*
- *Template Function mir::operator<=*
- *Template Function mir::operator==(optional_value<T> const&, optional_value<T> const&)*
- *Template Function mir::operator==(IntWrapper<Tag, ValueType> const&, IntWrapper<Tag, ValueType> const&)*
- *Template Function mir::operator==(optional_value<T> const&, T const&)*
- *Template Function mir::operator==(T const&, optional_value<T> const&)*
- *Template Function mir::operator>=*

Typedefs

- *Typedef mir::EventUPtr*

Variables

- *Variable mir::fatal_error*

Namespace mir::geometry

Basic geometry types. Types for dimensions, displacements, etc. and the operations that they support.

Namespaces

- *Namespace mir::geometry::generic*

Classes

- *Struct DeltaXTag*
- *Struct DeltaYTag*
- *Struct HeightTag*
- *Struct StrideTag*
- *Struct WidthTag*
- *Struct XTag*
- *Struct YTag*
- *Class Rectangles*

Functions

- *Template Function mir::geometry::as_delta(generic::Y<T> const&)*
- *Template Function mir::geometry::as_delta(generic::Width<T> const&)*
- *Template Function mir::geometry::as_delta(generic::Height<T> const&)*
- *Template Function mir::geometry::as_delta(generic::X<T> const&)*
- *Template Function mir::geometry::as_height(generic::Y<T> const&)*
- *Template Function mir::geometry::as_height(generic::DeltaY<T> const&)*
- *Template Function mir::geometry::as_width(generic::X<T> const&)*
- *Template Function mir::geometry::as_width(generic::DeltaX<T> const&)*
- *Template Function mir::geometry::as_x(generic::Width<T> const&)*
- *Template Function mir::geometry::as_x(generic::DeltaX<T> const&)*
- *Template Function mir::geometry::as_y(generic::DeltaY<T> const&)*
- *Template Function mir::geometry::as_y(generic::Height<T> const&)*
- *Function mir::geometry::operator<<*

Typedefs

- *Typedef mir::geometry::DeltaX*
- *Typedef mir::geometry::DeltaXF*
- *Typedef mir::geometry::DeltaY*
- *Typedef mir::geometry::DeltaYF*
- *Typedef mir::geometry::Displacement*
- *Typedef mir::geometry::DisplacementF*
- *Typedef mir::geometry::Height*
- *Typedef mir::geometry::HeightF*
- *Typedef mir::geometry::Point*
- *Typedef mir::geometry::PointF*
- *Typedef mir::geometry::Rectangle*
- *Typedef mir::geometry::RectangleF*
- *Typedef mir::geometry::Size*
- *Typedef mir::geometry::SizeF*
- *Typedef mir::geometry::Stride*
- *Typedef mir::geometry::Width*
- *Typedef mir::geometry::WidthF*
- *Typedef mir::geometry::X*
- *Typedef mir::geometry::XF*

- *Typedef mir::geometry::Y*
- *Typedef mir::geometry::YF*

Namespace mir::geometry::generic

Classes

- *Template Struct Displacement*
- *Template Struct Point*
- *Template Struct Rectangle*
- *Template Struct Size*
- *Template Struct Value*

Functions

- *Template Function mir::geometry::generic::as_displacement(Size<T> const&)*
- *Template Function mir::geometry::generic::as_displacement(Point<T> const&)*
- *Template Function mir::geometry::generic::as_point(Size<T> const&)*
- *Template Function mir::geometry::generic::as_point(Displacement<T> const&)*
- *Template Function mir::geometry::generic::as_size(Displacement<T> const&)*
- *Template Function mir::geometry::generic::as_size(Point<T> const&)*
- *Template Function mir::geometry::generic::intersection_of*
- *Template Function mir::geometry::generic::operator!=(Point<T> const&, Point<T> const&)*
- *Template Function mir::geometry::generic::operator!=(Displacement<T> const&, Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator!=(Rectangle<T> const&, Rectangle<T> const&)*
- *Template Function mir::geometry::generic::operator!=(Size<T> const&, Size<T> const&)*
- *Template Function mir::geometry::generic::operator*(Displacement<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator*(Scalar, DeltaX<T> const&)*
- *Template Function mir::geometry::generic::operator*(Scalar, Size<T> const&)*
- *Template Function mir::geometry::generic::operator*(Width<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator*(Scalar, Width<T> const&)*
- *Template Function mir::geometry::generic::operator*(DeltaY<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator*(DeltaX<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator*(Scalar, Height<T> const&)*
- *Template Function mir::geometry::generic::operator*(Height<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator*(Size<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator*(Scalar, DeltaY<T> const&)*
- *Template Function mir::geometry::generic::operator*(Scalar, Displacement<T> const&)*

- *Template Function mir::geometry::generic::operator+(Height<T>, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator+(Point<T>, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator+(Height<T>, Height<T>)*
- *Template Function mir::geometry::generic::operator+(DeltaX<T>, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator+(X<T>, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator+(Displacement<T> const&, Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator+(Displacement<T> const&, Point<T> const&)*
- *Template Function mir::geometry::generic::operator+(Width<T>, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator+(DeltaY<T>, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator+(Width<T>, Width<T>)*
- *Template Function mir::geometry::generic::operator+(Y<T>, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator+(Point<T> const&, Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator+(Point<T>, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator+=(Width<T>&, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator+=(Point<T>&, Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator+=(Point<T>&, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator+=(Height<T>&, Height<T>)*
- *Template Function mir::geometry::generic::operator+=(DeltaX<T>&, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator+=(Width<T>&, Width<T>)*
- *Template Function mir::geometry::generic::operator+=(X<T>&, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator+=(Point<T>&, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator+=(DeltaY<T>&, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator+=(Y<T>&, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator+=(Height<T>&, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator-(Point<T> const&, Point<T> const&)*
- *Template Function mir::geometry::generic::operator-(Point<T> const&, Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator-(Width<T>, Width<T>)*
- *Template Function mir::geometry::generic::operator-(X<T>, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator-(DeltaX<T>)*
- *Template Function mir::geometry::generic::operator-(X<T>, X<T>)*
- *Template Function mir::geometry::generic::operator-(Point<T>, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator-(Height<T>, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator-(Width<T>, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator-(DeltaY<T>)*
- *Template Function mir::geometry::generic::operator-(Y<T>, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator-(DeltaY<T>, DeltaY<T>)*

- *Template Function mir::geometry::generic::operator-(Height<T>, Height<T>)*
- *Template Function mir::geometry::generic::operator-(Point<T>, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator-(Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator-(Y<T>, Y<T>)*
- *Template Function mir::geometry::generic::operator-(Displacement<T> const&, Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator-(DeltaX<T>, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator-=(Height<T>&, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator-=(Width<T>&, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator-=(Point<T>&, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator-=(Point<T>&, Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator-=(Y<T>&, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator-=(X<T>&, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator-=(DeltaY<T>&, DeltaY<T>)*
- *Template Function mir::geometry::generic::operator-=(DeltaX<T>&, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator-=(Point<T>&, DeltaX<T>)*
- *Template Function mir::geometry::generic::operator/(DeltaX<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator/(Size<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator/(Width<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator/(DeltaY<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator/(Height<T> const&, Scalar)*
- *Template Function mir::geometry::generic::operator<*
- *Template Function mir::geometry::generic::operator<<(std::ostream&, Rectangle<T> const&)*
- *Template Function mir::geometry::generic::operator<<(std::ostream&, Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator<<(std::ostream&, Value<T, Tag> const&)*
- *Template Function mir::geometry::generic::operator<<(std::ostream&, Point<T> const&)*
- *Template Function mir::geometry::generic::operator<<(std::ostream&, Size<T> const&)*
- *Template Function mir::geometry::generic::operator==(Size<T> const&, Size<T> const&)*
- *Template Function mir::geometry::generic::operator==(Displacement<T> const&, Displacement<T> const&)*
- *Template Function mir::geometry::generic::operator==(Point<T> const&, Point<T> const&)*
- *Template Function mir::geometry::generic::operator==(Rectangle<T> const&, Rectangle<T> const&)*

Typedefs

- *Typedef mir::geometry::generic::DeltaX*
- *Typedef mir::geometry::generic::DeltaY*
- *Typedef mir::geometry::generic::Height*
- *Typedef mir::geometry::generic::Width*
- *Typedef mir::geometry::generic::X*
- *Typedef mir::geometry::generic::Y*

Namespace miral

Mir Abstraction Layer.

Detailed Description

A thin, hopefully ABI stable, layer over the Mir libraries exposing only those abstractions needed to write a shell. One day this may inspire a core Mir library.

Namespaces

- *Namespace miral::detail*
- *Namespace miral::toolkit*

Classes

- *Struct ApplicationInfo*
- *Struct FdHandle*
- *Struct Output::PhysicalSizeMM*
- *Struct WaylandExtensions::Builder*
- *Struct WindowInfo*
- *Struct WindowManagerOption*
- *Struct WindowSpecification::AspectRatio*
- *Class AddInitCallback*
- *Class AppendEventFilter*
- *Class ApplicationAuthorizer*
- *Class ApplicationCredentials*
- *Class BasicSetApplicationAuthorizer*
- *Class CanonicalWindowManagerPolicy*
- *Class ConfigurationOption*
- *Class CursorTheme*

- *Class DisplayConfiguration*
- *Class ExternalClientLauncher*
- *Class InternalClientLauncher*
- *Class Keymap*
- *Class MinimalWindowManager*
- *Class MirRunner*
- *Class Output*
- *Class PrependEventFilter*
- *Template Class SetApplicationAuthorizer*
- *Class SetCommandLineHandler*
- *Class SetTerminator*
- *Class SetWindowManagementPolicy*
- *Class StartupInternalClient*
- *Class WaylandExtensions*
- *Class WaylandExtensions::Context*
- *Class WaylandExtensions::EnableInfo*
- *Class Window*
- *Class WindowManagementPolicy*
- *Class WindowManagerOptions*
- *Class WindowManagerTools*
- *Class WindowSpecification*
- *Class X11Support*
- *Class Zone*

Functions

- *Template Function miral::add_window_manager_policy*
- *Function miral::application_for(wl_client *)*
- *Function miral::application_for(wl_resource *)*
- *Function miral::apply_lifecycle_state_to*
- *Function miral::display_configuration_options*
- *Function miral::equivalent_display_area*
- *Function miral::kill*
- *Template Function miral::lambda_as_function*
- *Function miral::name_of*
- *Function miral::operator!=(std::shared_ptr<mir::scene::Surface> const&, Window const&)*
- *Function miral::operator!=(Output::PhysicalSizeMM const&, Output::PhysicalSizeMM const&)*

- *Function miral::operator!=(Window const&, std::shared_ptr<mir::scene::Surface> const&)*
- *Function miral::operator!=(Window const&, Window const&)*
- *Function miral::operator<*
- *Function miral::operator<=*
- *Function miral::operator==(std::shared_ptr<mir::scene::Surface> const&, Window const&)*
- *Function miral::operator==(Window const&, std::shared_ptr<mir::scene::Surface> const&)*
- *Function miral::operator==(Output::PhysicalSizeMM const&, Output::PhysicalSizeMM const&)*
- *Function miral::operator==(Window const&, Window const&)*
- *Function miral::operator>*
- *Function miral::operator>=*
- *Function miral::pid_of*
- *Function miral::pre_init*
- *Function miral::PrintTo*
- *Template Function miral::set_window_management_policy*
- *Function miral::socket_fd_of*
- *Function miral::window_for*

Typedefs

- *Typedef miral::Application*
- *Typedef miral::BufferStreamId*
- *Typedef miral::CommandLineOption*
- *Typedef miral::WindowManagementPolicyBuilder*

Namespace miral::detail

Classes

- *Template Struct FunctionType*
- *Template Struct FunctionType< Return(Lambda::*)(Arg…) const >*
- *Template Struct FunctionType< Return(Lambda::*)(Arg…) >*

Namespace miral::toolkit

Functions

- *Function miral::toolkit::mir_event_get_input_event*
- *Function miral::toolkit::mir_event_get_type*
- *Function miral::toolkit::mir_input_event_get_event*
- *Function miral::toolkit::mir_input_event_get_event_time*
- *Function miral::toolkit::mir_input_event_get_keyboard_event*
- *Function miral::toolkit::mir_input_event_get_pointer_event*
- *Function miral::toolkit::mir_input_event_get_touch_event*
- *Function miral::toolkit::mir_input_event_get_type*
- *Function miral::toolkit::mir_input_event_has_cookie*
- *Function miral::toolkit::mir_keyboard_event_action*
- *Function miral::toolkit::mir_keyboard_event_input_event*
- *Function miral::toolkit::mir_keyboard_event_key_text*
- *Function miral::toolkit::mir_keyboard_event_keysym*
- *Function miral::toolkit::mir_keyboard_event_modifiers*
- *Function miral::toolkit::mir_keyboard_event_scan_code*
- *Function miral::toolkit::mir_pointer_event_action*
- *Function miral::toolkit::mir_pointer_event_axis_value*
- *Function miral::toolkit::mir_pointer_event_button_state*
- *Function miral::toolkit::mir_pointer_event_buttons*
- *Function miral::toolkit::mir_pointer_event_input_event*
- *Function miral::toolkit::mir_pointer_event_modifiers*
- *Function miral::toolkit::mir_touch_event_action*
- *Function miral::toolkit::mir_touch_event_axis_value*
- *Function miral::toolkit::mir_touch_event_id*
- *Function miral::toolkit::mir_touch_event_input_event*
- *Function miral::toolkit::mir_touch_event_modifiers*
- *Function miral::toolkit::mir_touch_event_point_count*
- *Function miral::toolkit::mir_touch_event_tooltype*

Namespace miroil

Classes

- *Struct DisplayConfigurationOptions*
- *Struct DisplayConfigurationOptions::DisplayMode*
- *Struct DisplayId*
- *Struct Edid*
- *Struct Edid::Descriptor*
- *Struct Edid::PhysicalSizeMM*
- *Class Compositor*
- *Class DisplayConfigurationControllerWrapper*
- *Class DisplayConfigurationPolicy*
- *Class DisplayConfigurationStorage*
- *Class DisplayListenerWrapper*
- *Class EventBuilder*
- *Class EventBuilder::EventInfo*
- *Class GLBuffer*
- *Class InputDevice*
- *Class InputDeviceObserver*
- *Class MirPromptSession*
- *Class MirServerHooks*
- *Class OpenGLContext*
- *Class PersistDisplayConfig*
- *Class PromptSessionListener*
- *Class PromptSessionManager*
- *Class SetCompositor*
- *Class Surface*
- *Class SurfaceObserver*

Functions

- *Function miroil::dispatch_input_event*

Typedefs

- *Typedef miroil::CompositorID*
- *Typedef miroil::CreateNamedCursor*
- *Typedef miroil::OutputId*

Unions

- *Union Descriptor::Value*

Namespace std

STL namespace.

Classes

- *Template Struct hash< ::mir::IntWrapper< Tag, ValueType > >*

Functions

- *Specialized Template Function std::swap*

Namespace wallpaper

Functions

- *Function wallpaper::font_file()*
- *Function wallpaper::font_file(std::string const&)*

Classes and Structs

Struct DeltaXTag

- Defined in file_include_core_mir_geometry_forward.h

Struct Documentation

struct **DeltaXTag**

Struct DeltaYTag

- Defined in file_include_core_mir_geometry_forward.h

Struct Documentation

struct **DeltaYTag**

Template Struct Displacement

- Defined in file_include_core_mir_geometry_displacement.h

Struct Documentation

template<typename **T**>

struct **Displacement**

Public Types

using **ValueType** = *T*

Public Functions

inline constexpr **Displacement**()

constexpr **Displacement**(*Displacement* const&) = default

Displacement &**operator=**(*Displacement* const&) = default

template<typename **U**>
inline explicit constexpr **Displacement**(*Displacement*<*U*> const &other) noexcept

template<typename **DeltaXType**, typename **DeltaYType**>
inline constexpr **Displacement**(*DeltaXType* &&dx, *DeltaYType* &&dy)

template<typename **Q** = *T*>
inline constexpr std::enable_if<std::is_integral<*Q*>::value, long long>::type **length_squared**() const

template<typename **Q** = *T*>
inline constexpr std::enable_if<!std::is_integral<*Q*>::value, *T*>::type **length_squared**() const

Public Members

DeltaX<*T*> **dx**

DeltaY<*T*> **dy**

Template Struct Point

- Defined in file_include_core_mir_geometry_point.h

Struct Documentation

template<typename **T**>

struct **Point**

Public Types

using **ValueType** = *T*

Public Functions

constexpr **Point**() = default

constexpr **Point**(*Point* const&) = default

Point &**operator=**(*Point* const&) = default

template<typename **U**>
inline explicit constexpr **Point**(*Point*<*U*> const &other) noexcept

template<typename **XType**, typename **YType**>
inline constexpr **Point**(*XType* &&x, *YType* &&y)

Public Members

X<*T*> **x**

Y<*T*> **y**

Template Struct Rectangle

- Defined in file_include_core_mir_geometry_rectangle.h

Struct Documentation

template<typename **T**>

struct **Rectangle**

Public Functions

constexpr **Rectangle**() = default

inline constexpr **Rectangle**(*Point*<*T*> const &top_left, *Size*<*T*> const &size)

inline *Point*<*T*> **bottom_right**() const

The bottom right boundary point of the rectangle.

Note that the returned point is *not* included in the rectangle area, that is, the rectangle is represented as [top_left,bottom_right).

inline *Point*<*T*> **top_right**() const

inline *Point*<*T*> **bottom_left**() const

inline bool **contains**(*Point*<*T*> const &p) const

inline bool **contains**(*Rectangle*<*T*> const &r) const

Test if the rectangle contains another.

Note that an empty rectangle can still contain other empty rectangles, which are treated as points or lines of thickness zero.

inline bool **overlaps**(*Rectangle*<*T*> const &r) const

inline *X*<*T*> **left**() const

inline *X*<*T*> **right**() const

inline *Y*<*T*> **top**() const

inline *Y*<*T*> **bottom**() const

Public Members

Point<*T*> **top_left**

Size<*T*> **size**

Template Struct Size

- Defined in file_include_core_mir_geometry_size.h

Struct Documentation

template<typename **T**>

struct **Size**

Public Types

using **ValueType** = *T*

Public Functions

inline constexpr **Size**() noexcept

constexpr **Size**(*Size* const&) noexcept = default

Size &**operator=**(*Size* const&) noexcept = default

template<typename **U**>
inline explicit constexpr **Size**(*Size*<*U*> const &other) noexcept

template<typename **WidthType**, typename **HeightType**>
inline constexpr **Size**(*WidthType* &&width, *HeightType* &&height) noexcept

Public Members

Width<*T*> **width**

Height<*T*> **height**

Template Struct Value

- Defined in file_include_core_mir_geometry_dimensions.h

Struct Documentation

template<typename **T**, typename **Tag**>

struct **Value**

Wraps a geometry value and prevents it from being accidentally used for invalid operations (such as setting a width to a height or adding two x positions together). Of course, explicit casts are possible to get around these restrictions (see the as_*() functions).

Public Types

using **ValueType** = *T*

using **TagType** = *Tag*

Public Functions

template<typename **Q** = *T*>
inline constexpr std::enable_if<std::is_integral<*Q*>::value, int>::type **as_int**() const

template<typename **Q** = *T*>
inline constexpr std::enable_if<std::is_integral<*Q*>::value, uint32_t>::type **as_uint32_t**() const

inline constexpr *T* **as_value**() const noexcept

inline constexpr **Value**() noexcept

inline *Value* &**operator=**(*Value* const &that) noexcept

inline constexpr **Value**(*Value* const &that) noexcept

template<typename **U**>
inline explicit constexpr **Value**(*Value*<*U*, *Tag*> const &value) noexcept

template<typename **U**, typename std::enable_if<std::is_scalar<*U*>::value, bool>::type = true>
inline explicit constexpr **Value**(*U* const &value) noexcept

inline constexpr auto **operator==**(*Value*<*T*, *Tag*> const &rhs) const -> bool

inline constexpr auto **operator!=**(*Value*<*T*, *Tag*> const &rhs) const -> bool

inline constexpr auto **operator<=**(*Value*<*T*, *Tag*> const &rhs) const -> bool

inline constexpr auto **operator>=**(*Value*<*T*, *Tag*> const &rhs) const -> bool

inline constexpr auto **operator<**(*Value*<*T*, *Tag*> const &rhs) const -> bool

inline constexpr auto **operator>**(*Value*<*T*, *Tag*> const &rhs) const -> bool

Protected Attributes

T **value**

Struct HeightTag

- Defined in file_include_core_mir_geometry_forward.h

Struct Documentation

struct **HeightTag**

Struct StrideTag

- Defined in file_include_core_mir_geometry_forward.h

Struct Documentation

struct **StrideTag**

Struct WidthTag

- Defined in file_include_core_mir_geometry_forward.h

Struct Documentation

struct **WidthTag**

These tag types determine what type of dimension a value holds and what operations are possible with it. They are only used as template parameters, are never instantiated and should only require forward declarations, but some compiler versions seem to fail if they aren't given real declarations.

Struct XTag

- Defined in file_include_core_mir_geometry_forward.h

Struct Documentation

struct **XTag**

Struct YTag

- Defined in file_include_core_mir_geometry_forward.h

Struct Documentation

struct **YTag**

Struct IntOwnedFd

- Defined in file_include_core_mir_fd.h

Struct Documentation

struct **IntOwnedFd**

Public Members

int **int_owned_fd**

Struct ApplicationInfo

- Defined in file_include_miral_miral_application_info.h

Struct Documentation

struct **ApplicationInfo**

Public Functions

ApplicationInfo()

explicit **ApplicationInfo**(*Application* const &app)

~ApplicationInfo()

ApplicationInfo(*ApplicationInfo* const &that)

auto **operator=**(*ApplicationInfo* const &that) -> miral::*ApplicationInfo*&

auto **name**() const -> std::string

auto **application**() const -> *Application*

auto **windows**() const -> std::vector<*Window*>&

auto **userdata**() const -> std::shared_ptr<void>

This can be used by client code to store window manager specific information.

void **userdata**(std::shared_ptr<void> userdata)

Template Struct FunctionType

- Defined in file_include_miral_miral_lambda_as_function.h

Struct Documentation

template<class **F**>

struct **FunctionType**

Template Struct FunctionType< Return(Lambda::*)(Arg...) const >

- Defined in file_include_miral_miral_lambda_as_function.h

Struct Documentation

template<typename **Lambda**, typename **Return**, typename ...**Arg**>

struct **FunctionType**<*Return* (*Lambda*::*)(*Arg*...) const>

Public Types

using **type** = std::function<*Return*(*Arg*...)>

Template Struct FunctionType< Return(Lambda::*)(Arg...) >

- Defined in file_include_miral_miral_lambda_as_function.h

Struct Documentation

template<typename **Lambda**, typename **Return**, typename ...**Arg**>

struct **FunctionType**<*Return* (*Lambda*::*)(*Arg*...)>

Public Types

using **type** = std::function<*Return*(*Arg*...)>

Struct FdHandle

- Defined in file_include_miral_miral_runner.h

Struct Documentation

struct **FdHandle**

A handle which keeps a file descriptor registered to the main loop until it is dropped.

Public Functions

virtual **~FdHandle**()

Struct Output::PhysicalSizeMM

- Defined in file_include_miral_miral_output.h

Nested Relationships

This struct is a nested type of *Class Output*.

Struct Documentation

struct **PhysicalSizeMM**

Public Members

int **width**

int **height**

Struct WaylandExtensions::Builder

- Defined in file_include_miral_miral_wayland_extensions.h

Nested Relationships

This struct is a nested type of *Class WaylandExtensions*.

Struct Documentation

struct **Builder**

A *Builder* creates and registers an extension protocol.

Remark

Since MirAL 2.5

Public Members

std::string **name**

Name of the protocol extension's Wayland global.

std::function< std::shared_ptr< void >Context const *context)> build

Functor that creates and registers an extension protocol.

Param context

giving access to:

- the wl_display (so that, for example, the extension can be registered); and,
- allowing server initiated code to be executed on the Wayland mainloop.

Return

a shared pointer to the implementation. (Mir will manage the lifetime)

Struct WindowInfo

- Defined in file_include_miral_miral_window_info.h

Struct Documentation

struct **WindowInfo**

Unnamed Group

auto **min_width**() const -> mir::geometry::*Width*

These constrain the sizes a window may be resized to (both interactively and pragmatically). Clients can request a min/max size, but it can be overridden by the window management policy. By default, minimum values are 0 and maximum values are `std::numeric_limits<int>::max()`.

auto **min_height**() const -> mir::geometry::*Height*

auto **max_width**() const -> mir::geometry::*Width*

auto **max_height**() const -> mir::geometry::*Height*

Unnamed Group

auto **width_inc**() const -> mir::geometry::*DeltaX*

These control the size increments of the window. This is used in cases like a terminal that can only be resized character-by-character. Current Wayland protocols do not support this property, so it generally wont be requested by clients. By default, both are 1.

auto **height_inc**() const -> mir::geometry::*DeltaY*

Unnamed Group

auto **min_aspect**() const -> *AspectRatio*

These constrain the possible aspect ratio of the window. Current Wayland protocols to not support this property, so it generally wont be requested by clients. By default, min_aspect is `{0U, std::numeric_limits<unsigned>::max()}` and max_aspect is `{std::numeric_limits<unsigned>::max(), 0U}`.

auto **max_aspect**() const -> *AspectRatio*

Unnamed Group

auto **application_id**() const -> std::string

The D-bus service name and basename of the app's .desktop file See http://standards.freedesktop.org/desktop-entry-spec/.

Remark

Since MirAL 2.8

Public Types

using **AspectRatio** = *WindowSpecification*::*AspectRatio*

Public Functions

WindowInfo()

WindowInfo(*Window* const &window, *WindowSpecification* const &params)

~WindowInfo()

explicit **WindowInfo**(*WindowInfo* const &that)

WindowInfo &**operator=**(*WindowInfo* const &that)

bool **can_be_active**() const

bool **can_morph_to**(*MirWindowType* new_type) const

bool **must_have_parent**() const

bool **must_not_have_parent**() const

bool **is_visible**() const

void **constrain_resize**(mir::geometry::*Point* &requested_pos, mir::geometry::*Size* &requested_size) const

auto **window**() const -> *Window*&

auto **name**() const -> std::string

auto **type**() const -> *MirWindowType*

auto **state**() const -> *MirWindowState*

auto **restore_rect**() const -> mir::geometry::*Rectangle*

auto **parent**() const -> *Window*

auto **children**() const -> std::vector<*Window*> const&

bool **has_output_id**() const

auto **output_id**() const -> int

auto **preferred_orientation**() const -> *MirOrientationMode*

auto **confine_pointer**() const -> *MirPointerConfinementState*

auto **shell_chrome**() const -> *MirShellChrome*

auto **userdata**() const -> std::shared_ptr<void>

This can be used by client code to store window manager specific information.

void **userdata**(std::shared_ptr<void> userdata)

inline void **swap**(*WindowInfo* &rhs)

auto **depth_layer**() const -> *MirDepthLayer*

auto **attached_edges**() const -> *MirPlacementGravity*

Get the edges of the output that the window is attached to (only meaningful for windows in state mir_window_state_attached)

auto **exclusive_rect**() const -> mir::*optional_value*<mir::geometry::*Rectangle*>

Mir will try to avoid occluding the area covered by this rectangle (relative to the window) (only meaningful when the window is attached to an edge)

auto **clip_area**() const -> mir::*optional_value*<mir::geometry::*Rectangle*>

Mir will not render anything outside this rectangle.

void **clip_area**(mir::*optional_value*<mir::geometry::*Rectangle*> const &area)

auto **focus_mode**() const -> *MirFocusMode*

How the window should gain and lose focus.

Remark

Since MirAL 3.3

---

auto **visible_on_lock_screen**() const -> bool

If this surface should be shown while the compositor is locked.

Remark

Since MirAL 3.9

Public Static Functions

static bool **needs_titlebar**(*MirWindowType* type)

Struct WindowManagerOption

- Defined in file_include_miral_miral_window_management_options.h

Struct Documentation

struct **WindowManagerOption**

Public Members

std::string const **name**

WindowManagementPolicyBuilder const **build**

Struct WindowSpecification::AspectRatio

- Defined in file_include_miral_miral_window_specification.h

Nested Relationships

This struct is a nested type of *Class WindowSpecification*.

Struct Documentation

struct **AspectRatio**

Public Members

unsigned **width**

unsigned **height**

Struct MirBufferPackage

- Defined in file_include_core_mir_toolkit_mir_native_buffer.h

Struct Documentation

struct **MirBufferPackage**

Public Members

int **data_items**

int **fd_items**

int **data**[*mir_buffer_package_max*]

int **width**

int **height**

int **fd**[*mir_buffer_package_max*]

int **unused0**

unsigned int **flags**

int **stride**

int **age**

Number of frames submitted by the client since the client has rendered to this buffer.

This has the same semantics as the EGL_EXT_buffer_age extension

See also:

http://www.khronos.org/registry/egl/extensions/EXT/EGL_EXT_buffer_age.txt

Struct DisplayConfigurationOptions

- Defined in file_include_miroil_miroil_display_configuration_storage.h

Nested Relationships

Nested Types

- *Struct DisplayConfigurationOptions::DisplayMode*

Struct Documentation

struct **DisplayConfigurationOptions**

Public Members

mir::*optional_value*<bool> **used**

mir::*optional_value*<uint> **clone_output_index**

mir::*optional_value*<*DisplayMode*> **mode**

mir::*optional_value*<*MirOrientation*> **orientation**

mir::*optional_value*<*MirFormFactor*> **form_factor**

mir::*optional_value*<float> **scale**

struct **DisplayMode**

Public Members

mir::geometry::*Size* **size**

double **refresh_rate** = {-1}

Struct DisplayConfigurationOptions::DisplayMode

- Defined in file_include_miroil_miroil_display_configuration_storage.h

Nested Relationships

This struct is a nested type of *Struct DisplayConfigurationOptions*.

Struct Documentation

struct **DisplayMode**

Public Members

mir::geometry::*Size* **size**

double **refresh_rate** = {-1}

Struct DisplayId

- Defined in file_include_miroil_miroil_display_id.h

Struct Documentation

struct **DisplayId**

Public Members

Edid **edid**

OutputId **output_id**

Struct Edid

- Defined in file_include_miroil_miroil_edid.h

Nested Relationships

Nested Types

- *Struct Edid::Descriptor*
- *Union Descriptor::Value*
- *Struct Edid::PhysicalSizeMM*

Struct Documentation

struct **Edid**

Public Functions

Edid &**parse_data**(std::vector<uint8_t> const&)

Public Members

std::string **vendor**

uint16_t **product_code** = {0}

uint32_t **serial_number** = {0}

PhysicalSizeMM **size** = {0, 0}

Descriptor **descriptors**[4]

struct **Descriptor**

Public Types

enum class **Type** : uint8_t

Values:

enumerator **timing_identifiers**

enumerator **white_point_data**

enumerator **monitor_name**

enumerator **monitor_limits**

enumerator **unspecified_text**

enumerator **serial_number**

enumerator **undefined**

Public Functions

std::string **string_value**() const

Public Members

Type **type** = {*Type*::*undefined*}

Value **value** = {{0}}

union **Value**

Public Members

char **monitor_name**[13]

char **unspecified_text**[13]

char **serial_number**[13]

struct **PhysicalSizeMM**

Public Members

int **width**

int **height**

Struct Edid::Descriptor

- Defined in file_include_miroil_miroil_edid.h

Nested Relationships

This struct is a nested type of *Struct Edid*.

Nested Types

- *Union Descriptor::Value*

Struct Documentation

struct **Descriptor**

Public Types

enum class **Type** : uint8_t

Values:

enumerator **timing_identifiers**

enumerator **white_point_data**

enumerator **monitor_name**

enumerator **monitor_limits**

enumerator **unspecified_text**

enumerator **serial_number**

enumerator **undefined**

Public Functions

std::string **string_value**() const

Public Members

Type **type** = {*Type*::*undefined*}

Value **value** = {{0}}

union **Value**

Public Members

char **monitor_name**[13]

char **unspecified_text**[13]

char **serial_number**[13]

Struct Edid::PhysicalSizeMM

- Defined in file_include_miroil_miroil_edid.h

Nested Relationships

This struct is a nested type of *Struct Edid*.

Struct Documentation

struct **PhysicalSizeMM**

Public Members

int **width**

int **height**

Template Struct hash< ::mir::IntWrapper< Tag, ValueType > >

- Defined in file_include_core_mir_int_wrapper.h

Struct Documentation

template<typename **Tag**, typename **ValueType**>

struct **hash**<::mir::*IntWrapper*<*Tag*, *ValueType*>>

Public Functions

inline constexpr std::size_t **operator()**(::mir::*IntWrapper*<*Tag*, *ValueType*> const &id) const

Public Members

std::hash<int> **self**

Class DecorationProvider

- Defined in file_examples_example-server-lib_decoration_provider.h

Class Documentation

class **DecorationProvider**

Public Functions

DecorationProvider()

~DecorationProvider()

void **operator()**(struct wl_display *display)

void **operator()**(std::weak_ptr<mir::scene::Session> const &session)

auto **session**() const -> std::shared_ptr<mir::scene::Session>

void **stop**()

bool **is_decoration**(miral::*Window* const &window) const

Class FloatingWindowManagerPolicy

- Defined in file_examples_example-server-lib_floating_window_manager.h

Inheritance Relationships

Base Type

- public miral::MinimalWindowManager (*Class MinimalWindowManager*)

Class Documentation

class **FloatingWindowManagerPolicy** : public miral::*MinimalWindowManager*

example event handling:

o Switch apps: Alt+Tab, tap or click on the corresponding window o Switch window: Alt+`, tap or click on the corresponding window o Move window: Alt-leftmousebutton drag (three finger drag) o Resize window: Alt-middle_button drag (three finger pinch) o Maximize/restore current window (to display size): Alt-F11 o Maximize/restore current window (to display height): Shift-F11 o Maximize/restore current window (to display width): Ctrl-F11 o Switch workspace .

… … … : Meta-Alt-[F1|F2|F3|F4] o Switch workspace taking active window: Meta-Ctrl-[F1|F2|F3|F4]

virtual bool **handle_pointer_event**(MirPointerEvent const *event) override

pointer event handler

Parameters

event – the event

Returns
whether the policy has consumed the event

virtual bool **handle_touch_event**(MirTouchEvent const *event) override

touch event handler

Parameters
event – the event

Returns
whether the policy has consumed the event

virtual bool **handle_keyboard_event**(MirKeyboardEvent const *event) override

keyboard event handler

Parameters
event – the event

Returns
whether the policy has consumed the event

track events that affect titlebar

virtual void **advise_new_window**(miral::*WindowInfo* const &window_info) override

Notification that a window has been created.

Parameters
window_info – the window

virtual void **handle_window_ready**(miral::*WindowInfo* &window_info) override

notification that the first buffer has been posted

Parameters
window_info – the window

virtual void **advise_focus_gained**(miral::*WindowInfo* const &info) override

Notification that a window has gained focus.

Parameters
window_info – the window

virtual void **handle_modify_window**(miral::*WindowInfo* &window_info, miral::*WindowSpecification* const &modifications) override

request from client to modify the window specification.

Parameters

- **window_info** – the window
- **modifications** – the requested changes

Note: the request has already been validated against the type definition

---

Public Functions

FloatingWindowManagerPolicy(miral::*WindowManagerTools* const &tools, std::shared_ptr<*SplashSession*> const &spinner, miral::*InternalClientLauncher* const &launcher, std::function<void()> &shutdown_hook)

~FloatingWindowManagerPolicy()

virtual miral::*WindowSpecification* **place_new_window**(miral::*ApplicationInfo* const &app_info, miral::*WindowSpecification* const &request_parameters) override

Customize initial window placement.

Parameters

- **app_info** – the application requesting a new window
- **requested_specification** – the requested specification (updated with default placement)

Returns

the customized specification

Protected Static Attributes

static const int **modifier_mask** = mir_input_event_modifier_alt | mir_input_event_modifier_shift | mir_input_event_modifier_sym | mir_input_event_modifier_ctrl | mir_input_event_modifier_meta

Class KioskWindowManagerPolicy

- Defined in file_examples_miral-kiosk_kiosk_window_manager.h

Inheritance Relationships

Base Type

- `public miral::CanonicalWindowManagerPolicy` (*Class CanonicalWindowManagerPolicy*)

Class Documentation

class **KioskWindowManagerPolicy** : public miral::*CanonicalWindowManagerPolicy*

Public Functions

KioskWindowManagerPolicy(miral::*WindowManagerTools* const &tools, std::shared_ptr<*SplashSession*> const&)

virtual auto **place_new_window**(miral::*ApplicationInfo* const &app_info, miral::*WindowSpecification* const &request) -> miral::*WindowSpecification* override

Customize initial window placement.

Parameters

- **app_info** – the application requesting a new window
- **requested_specification** – the requested specification (updated with default placement)

Returns

the customized specification

virtual void **advise_focus_gained**(miral::*WindowInfo* const &info) override

Notification that a window has gained focus.

Parameters

window_info – the window

virtual bool **handle_keyboard_event**(MirKeyboardEvent const *event) override

keyboard event handler

Parameters

event – the event

Returns

whether the policy has consumed the event

virtual bool **handle_touch_event**(MirTouchEvent const *event) override

touch event handler

Parameters

event – the event

Returns

whether the policy has consumed the event

virtual bool **handle_pointer_event**(MirPointerEvent const *event) override

pointer event handler

Parameters

event – the event

Returns

whether the policy has consumed the event

virtual void **handle_modify_window**(miral::*WindowInfo* &window_info, miral::*WindowSpecification* const &modifications) override

request from client to modify the window specification.

Parameters

- **window_info** – the window
- **modifications** – the requested changes

Note: the request has already been validated against the type definition

---

virtual void **handle_request_move**(miral::*WindowInfo* &window_info, MirInputEvent const *input_event) override

request from client to initiate move

Parameters

- **window_info** – the window
- **input_event** – the requesting event

Note: the request has already been validated against the requesting event

virtual void **handle_request_resize**(miral::*WindowInfo* &window_info, MirInputEvent const *input_event, *MirResizeEdge* edge) override

request from client to initiate resize

Parameters

- **window_info** – the window
- **input_event** – the requesting event
- **edge** – the edge(s) being dragged

Note: the request has already been validated against the requesting event

virtual Rectangle **confirm_placement_on_display**(const miral::*WindowInfo* &window_info, *MirWindowState* new_state, Rectangle const &new_placement) override

Confirm (and optionally adjust) the placement of a window on the display.

Called when (re)placing fullscreen, maximized, horizontally maximised and vertically maximized windows to allow adjustment for decorations.

Parameters

- **window_info** – the window
- **new_state** – the new state
- **new_placement** – the suggested placement

Returns

the confirmed placement of the window

Class AbnormalExit

- Defined in file_include_core_mir_abnormal_exit.h

Inheritance Relationships

Base Type

- `public std::runtime_error`

Derived Type

- `public mir::ExitWithOutput` (*Class ExitWithOutput*)

Class Documentation

class **AbnormalExit** : public std::runtime_error

Indicates Mir should exit with an error message printed to stderr.

Subclassed by *mir::ExitWithOutput*

Public Functions

inline **AbnormalExit**(std::string const &what)

Class AnonymousShmFile

- Defined in file_include_core_mir_anonymous_shm_file.h

Inheritance Relationships

Base Type

- `public mir::ShmFile` (*Class ShmFile*)

Class Documentation

class **AnonymousShmFile** : public mir::*ShmFile*

Public Functions

AnonymousShmFile(size_t size)

~AnonymousShmFile() noexcept

virtual void ***base_ptr**() const override

virtual int **fd**() const override

Class ExitWithOutput

- Defined in file_include_core_mir_abnormal_exit.h

Inheritance Relationships

Base Type

- `public mir::AbnormalExit` (*Class AbnormalExit*)

Class Documentation

class **ExitWithOutput** : public mir::*AbnormalExit*

Indicates Mir should exit with the given output (such as help text) printed to stdout.

Public Functions

inline **ExitWithOutput**(std::string const &what)

Class FatalErrorStrategy

- Defined in file_include_core_mir_fatal.h

Class Documentation

class **FatalErrorStrategy**

Public Functions

inline explicit **FatalErrorStrategy**(void (*fatal_error_handler)(char const *reason, ...))

inline **~FatalErrorStrategy**()

Class Fd

- Defined in file_include_core_mir_fd.h

Class Documentation

class **Fd**

Public Functions

explicit **Fd**(int fd)

explicit **Fd**(*IntOwnedFd*)

Fd()

Fd(*Fd*&&)

Fd(*Fd* const&) = default

Fd &**operator=**(*Fd*)

operator int() const

Public Static Attributes

static int const **invalid** = {-1}

Friends

friend auto **close**(*Fd* const &fd) -> int = delete

Prevent accidental calling of ::close()

Class Rectangles

- Defined in file_include_core_mir_geometry_rectangles.h

Class Documentation

class **Rectangles**

A collection of rectangles (with possible duplicates).

Public Types

typedef std::vector<*Rectangle*>::const_iterator **const_iterator**

typedef std::vector<*Rectangle*>::size_type **size_type**

Public Functions

Rectangles()

Rectangles(std::initializer_list<*Rectangle*> const &rects)

void **add**(*Rectangle* const &rect)

void **remove**(*Rectangle* const &rect)
removes at most one matching rectangle

void **clear**()

Rectangle **bounding_rectangle**() const

void **confine**(*Point* &point) const

const_iterator **begin**() const

const_iterator **end**() const

size_type **size**() const

bool **operator==**(*Rectangles* const &rect) const

bool **operator!=**(*Rectangles* const &rect) const

Template Class IntWrapper

- Defined in file_include_core_mir_int_wrapper.h

Class Documentation

template<typename **Tag**, typename **ValueType** = int>

class **IntWrapper**

Public Functions

inline constexpr **IntWrapper**()

inline explicit constexpr **IntWrapper**(*ValueType* value)

inline constexpr *ValueType* **as_value**() const

Template Class optional_value

- Defined in file_include_core_mir_optional_value.h

Class Documentation

template<typename **T**>

class **optional_value**

Public Functions

optional_value() = default

inline **optional_value**(*T* const &value)

inline *optional_value* &**operator=**(*T* const &value)

inline bool **is_set**() const

inline *T* const &**value**() const

inline *T* &**value**()

inline *T* &&**consume**()

inline **operator bool**() const

Class ProofOfMutexLock

- Defined in file_include_core_mir_proof_of_mutex_lock.h

Class Documentation

class **ProofOfMutexLock**

A method can take an instance of this class by reference to require callers to hold a mutex lock, without requiring a specific type of lock.

Public Functions

inline **ProofOfMutexLock**(std::lock_guard<std::mutex> const&)

inline **ProofOfMutexLock**(std::unique_lock<std::mutex> const &lock)

ProofOfMutexLock(*ProofOfMutexLock* const&) = delete

ProofOfMutexLock **operator=**(*ProofOfMutexLock* const&) = delete

Class ShmFile

- Defined in file_include_core_mir_shm_file.h

Inheritance Relationships

Derived Type

- `public mir::AnonymousShmFile` (*Class AnonymousShmFile*)

Class Documentation

class **ShmFile**

Subclassed by *mir::AnonymousShmFile*

Public Functions

virtual **~ShmFile**() = default

virtual void ***base_ptr**() const = 0

virtual int **fd**() const = 0

Protected Functions

ShmFile() = default

ShmFile(*ShmFile* const&) = delete

ShmFile &**operator=**(*ShmFile* const&) = delete

Template Class Synchronised

- Defined in file_include_core_mir_synchronised.h

Nested Relationships

Nested Types

- *Template Class Synchronised::LockedImpl*

Class Documentation

template<typename **T**>

class **Synchronised**

An object that enforces unique access to the data it contains.

This behaves like a mutex which owns the data it guards, and can give out a smart-pointer-esque handle to lock and access it.

Template Parameters

T – The type of data contained

Public Types

using **Locked** = *LockedImpl*<*T*>

Smart-pointer-esque accessor for the protected data.

Ensures exclusive access to the referenced data.

Note: Instances of Locked must not outlive the *Synchronised* they are derived from.

using **LockedView** = *LockedImpl*<*T* const>

Smart-pointer-esque accessor for the protected data.

Provides const access to the protected data, with the guarantee that no changes to the data can be made while this handle is live.

Note: Instances of Locked must not outlive the *Synchronised* they are derived from.

Public Functions

Synchronised() = default

inline **Synchronised**(*T* &&initial_value)

Synchronised(*Synchronised* const&) = delete

Synchronised &**operator=**(*Synchronised* const&) = delete

inline auto **lock**() -> *Locked*

Lock the data and return an accessor.

Returns

A smart-pointer-esque accessor for the contained data. While code has access to a live Locked instance it is guaranteed to have unique access to the contained data.

inline auto **lock**() const -> *LockedView*

Lock the data and return an accessor.

Returns

A smart-pointer-esque accessor for the contained data. While code has access to a live Locked instance it is guaranteed to have unique access to the contained data.

template<typename **U**>

class **LockedImpl**

Smart-pointer-esque accessor for the protected data.

Ensures exclusive access to the referenced data.

Note: Instances of Locked must not outlive the *Synchronised* they are derived from.

Public Functions

inline **LockedImpl**(*LockedImpl* &&from) noexcept

~LockedImpl() = default

inline auto **operator***() const -> *U*&

inline auto **operator->**() const -> *U**

inline void **drop**()

Relinquish access to the data.

This prevents further access to the contained data through this handle, and allows other code to acquire access.

template<typename **Cv**, typename **Predicate**>
inline void **wait**(*Cv* &cv, *Predicate* stop_waiting)

Allows waiting for a condition variable.

The protected data may be accessed both in the predicate and after this method completes.

Template Class Synchronised::LockedImpl

- Defined in file_include_core_mir_synchronised.h

Nested Relationships

This class is a nested type of *Template Class Synchronised*.

Class Documentation

template<typename **U**>

class **LockedImpl**

Smart-pointer-esque accessor for the protected data.

Ensures exclusive access to the referenced data.

Note: Instances of Locked must not outlive the *Synchronised* they are derived from.

Public Functions

inline **LockedImpl**(*LockedImpl* &&from) noexcept

~LockedImpl() = default

inline auto **operator***() const -> *U*&

inline auto **operator->**() const -> *U**

inline void **drop**()

Relinquish access to the data.

This prevents further access to the contained data through this handle, and allows other code to acquire access.

template<typename **Cv**, typename **Predicate**>
inline void **wait**(*Cv* &cv, *Predicate* stop_waiting)

Allows waiting for a condition variable.

The protected data may be accessed both in the predicate and after this method completes.

Class AddInitCallback

- Defined in file_include_miral_miral_add_init_callback.h

Class Documentation

class **AddInitCallback**

Add a callback to be invoked when the server has been initialized, but before it starts. If multiple callbacks are added they will be invoked in the sequence added.

Public Types

using **Callback** = std::function<void()>

Public Functions

explicit **AddInitCallback**(*Callback* const &callback)

~AddInitCallback()

void **operator()**(mir::Server &server) const

Class AppendEventFilter

- Defined in file_include_miral_miral_append_event_filter.h

Class Documentation

class **AppendEventFilter**

Public Functions

explicit **AppendEventFilter**(std::function<bool(*MirEvent* const *event)> const &filter)

Append an event filter (after any existing filters, including the window manager). The supplied filter should return true if and only if it handles the event as filters later in the list will not be called.

Remark

the filter return type changed to bool in MirAL 3.6

void **operator()**(mir::Server &server)

Class ApplicationAuthorizer

- Defined in file_include_miral_miral_application_authorizer.h

Class Documentation

class **ApplicationAuthorizer**

Public Functions

ApplicationAuthorizer() = default

virtual **~ApplicationAuthorizer**() = default

ApplicationAuthorizer(*ApplicationAuthorizer* const&) = delete

ApplicationAuthorizer &**operator=**(*ApplicationAuthorizer* const&) = delete

virtual bool **connection_is_allowed**(*ApplicationCredentials* const &creds) = 0

virtual bool **configure_display_is_allowed**(*ApplicationCredentials* const &creds) = 0

virtual bool **set_base_display_configuration_is_allowed**(*ApplicationCredentials* const &creds) = 0

virtual bool **screencast_is_allowed**(*ApplicationCredentials* const &creds) = 0

virtual bool **prompt_session_is_allowed**(*ApplicationCredentials* const &creds) = 0

virtual bool **configure_input_is_allowed**(*ApplicationCredentials* const &creds) = 0

virtual bool **set_base_input_configuration_is_allowed**(*ApplicationCredentials* const &creds) = 0

Class ApplicationCredentials

- Defined in file_include_miral_miral_application_authorizer.h

Class Documentation

class **ApplicationCredentials**

Public Functions

ApplicationCredentials(mir::frontend::SessionCredentials const &creds)

pid_t **pid**() const

uid_t **uid**() const

gid_t **gid**() const

Class BasicSetApplicationAuthorizer

- Defined in file_include_miral_miral_application_authorizer.h

Inheritance Relationships

Derived Type

- `public miral::SetApplicationAuthorizer< Policy >` (*Template Class SetApplicationAuthorizer*)

Class Documentation

class **BasicSetApplicationAuthorizer**

Subclassed by *miral::SetApplicationAuthorizer< Policy >*

Public Functions

explicit **BasicSetApplicationAuthorizer**(std::function<std::shared_ptr<*ApplicationAuthorizer*>()> const &builder)

~BasicSetApplicationAuthorizer()

void **operator()**(mir::Server &server)

auto **the_application_authorizer**() const -> std::shared_ptr<*ApplicationAuthorizer*>

Class CanonicalWindowManagerPolicy

- Defined in file_include_miral_miral_canonical_window_manager.h

Inheritance Relationships

Base Type

- `public miral::WindowManagementPolicy` (*Class WindowManagementPolicy*)

Derived Type

- `public KioskWindowManagerPolicy` (*Class KioskWindowManagerPolicy*)

Class Documentation

class **CanonicalWindowManagerPolicy** : public miral::*WindowManagementPolicy*

Widely accepted defaults for window management.

Subclassed by *KioskWindowManagerPolicy*

Public Functions

explicit **CanonicalWindowManagerPolicy**(*WindowManagerTools* const &tools)

virtual auto **place_new_window**(*ApplicationInfo* const &app_info, *WindowSpecification* const &request_parameters) -> *WindowSpecification* override

Customize initial window placement.

Parameters

- **app_info** – the application requesting a new window
- **requested_specification** – the requested specification (updated with default placement)

Returns

the customized specification

virtual void **handle_window_ready**(*WindowInfo* &window_info) override

Tries to focus on the newly ready window.

virtual void **handle_modify_window**(*WindowInfo* &window_info, *WindowSpecification* const &modifications) override

Applies the requested modifications.

virtual void **handle_raise_window**(*WindowInfo* &window_info) override

Tries to focus on the newly ready window.

virtual void **advise_focus_gained**(*WindowInfo* const &info) override

Raises the window (and any children)

virtual auto **confirm_inherited_move**(*WindowInfo* const &window_info, Displacement movement) -> Rectangle override

Move the child window with the parent.

virtual auto **confirm_placement_on_display**(*WindowInfo* const &window_info, *MirWindowState* new_state, Rectangle const &new_placement) -> Rectangle override

Confirm (and optionally adjust) the placement of a window on the display.

Called when (re)placing fullscreen, maximized, horizontally maximised and vertically maximized windows to allow adjustment for decorations.

Parameters

- **window_info** – the window
- **new_state** – the new state
- **new_placement** – the suggested placement

Returns

the confirmed placement of the window

Protected Attributes

WindowManagerTools **tools**

Class ConfigurationOption

- Defined in file_include_miral_miral_configuration_option.h

Class Documentation

class **ConfigurationOption**

Add a user configuration option to Mir's option handling. By default the callback will be invoked following Mir initialisation but prior to the server starting. The value supplied to the callback will come from the command line, environment variable, config file or the default.

Note: Except for re-ordering implied by "pre_init()" the callbacks will be invoked in the order supplied. \Remark: Renamed (from CommandLineOption) in MirAL 3.6

Public Functions

ConfigurationOption(std::function<void(int value)> callback, std::string const &option, std::string const &description, int default_value)

ConfigurationOption(std::function<void(double value)> callback, std::string const &option, std::string const &description, double default_value)

ConfigurationOption(std::function<void(std::string const &value)> callback, std::string const &option, std::string const &description, std::string const &default_value)

ConfigurationOption(std::function<void(std::string const &value)> callback, std::string const &option, std::string const &description, char const *default_value)

ConfigurationOption(std::function<void(bool value)> callback, std::string const &option, std::string const &description, bool default_value)

ConfigurationOption(std::function<void(mir::*optional_value*<int> const &value)> callback, std::string const &option, std::string const &description)

ConfigurationOption(std::function<void(mir::*optional_value*<std::string> const &value)> callback, std::string const &option, std::string const &description)

ConfigurationOption(std::function<void(mir::*optional_value*<bool> const &value)> callback, std::string const &option, std::string const &description)

ConfigurationOption(std::function<void(bool is_set)> callback, std::string const &option, std::string const &description)

ConfigurationOption(std::function<void(std::vector<std::string> const &values)> callback, std::string const &option, std::string const &description)

template<typename **Lambda**>
inline **ConfigurationOption**(*Lambda* &&callback, std::string const &option, std::string const &description)

void **operator()**(mir::Server &server) const

~ConfigurationOption()

ConfigurationOption(*ConfigurationOption* const&)

auto **operator=**(*ConfigurationOption* const&) -> *ConfigurationOption*&

Friends

friend auto **pre_init**(*ConfigurationOption* const &clo) -> *ConfigurationOption*

Update the option to be called back *before* Mir initialization starts.

Parameters

clo – the option

Class CursorTheme

- Defined in file_include_miral_miral_cursor_theme.h

Class Documentation

class **CursorTheme**

Load an X-cursor theme, either the supplied default, or through the —cursor-theme config option.

Public Functions

explicit **CursorTheme**(std::string const &theme)

Specify a default theme.

~CursorTheme()

void **operator()**(mir::Server &server) const

Class DisplayConfiguration

- Defined in file_include_miral_miral_display_configuration.h

Class Documentation

class **DisplayConfiguration**

Enable display configuration. The config file (*miral::MirRunner::display_config_file()*) is located via the XDG Base Directory Specification. Vis: ($XDG_CONFIG_HOME or $HOME/.config followed by $XDG_CONFIG_DIRS)

Remark

Since MirAL 2.4

Note: From MirAL 3.8 will monitor the configuration file or, if none found, for the creation of a file in $XDG_CONFIG_HOME or $HOME/.config. Changes to this file will be reloaded. In addition, the selected layout may be overridden using a corresponding file: display_config_file() + "-layout" which will also be monitored and changes reloaded

Public Functions

explicit **DisplayConfiguration**(*MirRunner* const &mir_runner)

auto **layout_option**() -> *ConfigurationOption*

Provide the default 'display-layout' configuration option.

void **select_layout**(std::string const &layout)

Select a layout from the configuration.

auto **list_layouts**() -> std::vector<std::string>

List all layouts found in the config file.

void **add_output_attribute**(std::string const &key)

Enable a custom output attribute in the .display YAML.

Remark

Since MirAL 3.8

void **operator()**(mir::Server &server) const

~DisplayConfiguration()

DisplayConfiguration(*DisplayConfiguration* const&)

auto **operator=**(*DisplayConfiguration* const&) -> *DisplayConfiguration*&

Class ExternalClientLauncher

- Defined in file_include_miral_miral_external_client.h

Class Documentation

class **ExternalClientLauncher**

Public Functions

ExternalClientLauncher()

~ExternalClientLauncher()

void **operator()**(mir::Server &server)

auto **launch**(std::vector<std::string> const &command_line) const -> pid_t

Launch with client environment configured for Wayland. If X11 is enabled, then DISPLAY will also be set accordingly.

Remark

Return type changed from void in MirAL 3.0

Returns

The pid of the process that was launched.

auto **launch_using_x11**(std::vector<std::string> const &command_line) const -> pid_t

If X11 is enabled, then launch with client environment configured for X11. For the occasions it is desired to coerce applications into using X11.

Remark

Return type changed from void in MirAL 3.0

Returns

The pid of the process that was launched (or -1 if X11 is not enabled)

void **snapcraft_launch**(std::string const &desktop_file) const

Use the proposed `desktop-entry` snap interface to launch another snap.

Remark

Since MirAL 3.0

---

auto **launch**(std::string const &command) const -> pid_t

Launch with client environment configured for Wayland. If X11 is enabled, then DISPLAY will also be set accordingly.

Remark

Since MirAL 3.6

Returns

The pid of the process that was launched.

Public Static Functions

static auto **split_command**(std::string const &command) -> std::vector<std::string>

Split out the tokens of a (possibly escaped) command.

Remark

Since MirAL 3.6

Class InternalClientLauncher

- Defined in file_include_miral_miral_internal_client.h

Class Documentation

class **InternalClientLauncher**

Public Functions

InternalClientLauncher()

~InternalClientLauncher()

void **operator()**(mir::Server &server)

void **launch**(std::function<void(struct ::wl_display *display)> const &wayland_fd,
std::function<void(std::weak_ptr<mir::scene::Session> const session)> const &connect_notification) const

template<typename **ClientObject**>
inline void **launch**(*ClientObject* &client_object) const

Class Keymap

- Defined in file_include_miral_miral_keymap.h

Class Documentation

class **Keymap**

Load a keymap.

Public Functions

Keymap()

Apply keymap from the config.

explicit **Keymap**(std::string const &keymap)

Specify a keymap. Format is:

```
<language>[+<variant>[+<options>]]
```

Options is a comma separated list. e.g. “gb” or “us+dvorak”

~Keymap()

Keymap(*Keymap* const &that)

auto **operator=**(*Keymap* const &rhs) -> *Keymap*&

void **operator()**(mir::Server &server) const

void **set_keymap**(std::string const &keymap)

Specify a new keymap.

Class MinimalWindowManager

- Defined in file_include_miral_miral_minimal_window_manager.h

Inheritance Relationships

Base Type

- `public miral::WindowManagementPolicy` (*Class WindowManagementPolicy*)

Derived Type

- `public FloatingWindowManagerPolicy` (*Class FloatingWindowManagerPolicy*)

Class Documentation

class **MinimalWindowManager** : public miral::*WindowManagementPolicy*

Minimal implementation of a floating window management policy.

Remark

Since MirAL 2.5

Subclassed by *FloatingWindowManagerPolicy*

Public Functions

explicit **MinimalWindowManager**(*WindowManagerTools* const &tools)

MinimalWindowManager(*WindowManagerTools* const &tools, *MirInputEventModifier* pointer_drag_modifier)

Allows shells to change the modifer used to identify a window drag gesture The default is mir_input_event_modifier_alt.

Remark

Since MirAL 3.7

~MinimalWindowManager()

virtual auto **place_new_window**(*ApplicationInfo* const &app_info, *WindowSpecification* const &requested_specification) -> *WindowSpecification* override

Honours the requested specification.

virtual void **handle_window_ready**(*WindowInfo* &window_info) override

If the window can have focus it is given focus.

virtual void **handle_modify_window**(*WindowInfo* &window_info, *WindowSpecification* const &modifications) override

Honours the requested modifications.

virtual void **handle_raise_window**(*WindowInfo* &window_info) override

Gives focus to the requesting window (tree)

virtual auto **confirm_placement_on_display**(*WindowInfo* const &window_info, *MirWindowState* new_state, Rectangle const &new_placement) -> Rectangle override

Honours the requested placement.

virtual bool **handle_keyboard_event**(MirKeyboardEvent const *event) override

Handles Alt-Tab, Alt-Grave and Alt-F4.

virtual bool **handle_touch_event**(MirTouchEvent const *event) override

Handles touch to focus.

virtual bool **handle_pointer_event**(MirPointerEvent const *event) override

Handles pre-existing move & resize gestures, plus click to focus.

virtual void **handle_request_move**(*WindowInfo* &window_info, MirInputEvent const *input_event) override

Initiates a move gesture (only implemented for pointers)

virtual void **handle_request_resize**(*WindowInfo* &window_info, MirInputEvent const *input_event, *MirResizeEdge* edge) override

Initiates a resize gesture (only implemented for pointers)

virtual auto **confirm_inherited_move**(*WindowInfo* const &window_info, Displacement movement) -> Rectangle override

Honours the requested movement.

virtual void **advise_focus_gained**(*WindowInfo* const &window_info) override

Raises newly focused window.

virtual void **advise_focus_lost**(*WindowInfo* const &window_info) override

Notification that a window has lost focus.

Parameters

window_info – the window

virtual void **advise_new_app**(miral::*ApplicationInfo* &app_info) override

Notification that a new application has connected.

Parameters

application – the application

virtual void **advise_delete_app**(miral::*ApplicationInfo* const &app_info) override

Notification that an application has disconnected.

Parameters

application – the application

Protected Functions

bool **begin_pointer_move**(*WindowInfo* const &window_info, MirInputEvent const *input_event)

bool **begin_pointer_resize**(*WindowInfo* const &window_info, MirInputEvent const *input_event, *MirResizeEdge* const &edge)

bool **begin_touch_move**(*WindowInfo* const &window_info, MirInputEvent const *input_event)

bool **begin_touch_resize**(*WindowInfo* const &window_info, MirInputEvent const *input_event, *MirResizeEdge* const &edge)

Protected Attributes

WindowManagerTools **tools**

Class MirRunner

- Defined in file_include_miral_miral_runner.h

Class Documentation

class **MirRunner**

Runner for applying initialization options to Mir.

Public Functions

MirRunner(int argc, char const *argv[])

MirRunner(int argc, char const *argv[], char const *config_file)

~MirRunner()

void **add_start_callback**(std::function<void()> const &start_callback)

Add a callback to be invoked when the server has started, If multiple callbacks are added they will be invoked in the sequence added.

void **add_stop_callback**(std::function<void()> const &stop_callback)

Add a callback to be invoked when the server is about to stop, If multiple callbacks are added they will be invoked in the reverse sequence added.

void **register_signal_handler**(std::initializer_list<int> signals, std::function<void(int)> const &handler)

Add signal handler to the server's main loop.

Remark

Since MirAL 3.7

auto **register_fd_handler**(mir::*Fd* fd, std::function<void(int)> const &handler) -> std::unique_ptr<miral::*FdHandle*>

Add a watch on a file descriptor. The handler will be triggered when there is data to read on the Fd.

Remark

Since MirAL 3.7

void **set_exception_handler**(std::function<void()> const &handler)

Set a handler for exceptions caught in *run_with()*. *run_with()* invokes handler() in catch (...) blocks before returning EXIT_FAILURE. Hence the exception can be re-thrown to retrieve type information. The default action is to call mir::report_exception(std::cerr)

auto **run_with**(std::initializer_list<std::function<void(::mir::Server&)>> options) -> int

Apply the supplied initialization options and run the Mir server.

Returns

EXIT_SUCCESS or EXIT_FAILURE according to whether the server ran successfully

---

Note: blocks until the Mir server exits

---

void **stop**()

Tell the Mir server to exit.

auto **config_file**() const -> std::string

Name of the .config file. The .config file is located via the XDG Base Directory Specification: $XDG_CONFIG_HOME or $HOME/.config followed by $XDG_CONFIG_DIRS Config file entries are long form (e.g. "x11-output=1200x720")

Remark

Since MirAL 2.4

---

auto **display_config_file**() const -> std::string

Name of the .display configuration file. The .display file is located via the XDG Base Directory Specification: $XDG_CONFIG_HOME or $HOME/.config followed by $XDG_CONFIG_DIRS Config file entries are long form (e.g. "x11-output=1200x720")

Remark

Since MirAL 2.4

---

auto **wayland_display**() const -> mir::*optional_value*<std::string>

Get the Wayland endpoint name (if any) usable as a $WAYLAND_DISPLAY value.

Remark

Since MirAL 2.8

---

auto **x11_display**() const -> mir::*optional_value*<std::string>

Get the X11 socket name (if any) usable as a $DISPLAY value.

Remark

Since MirAL 2.8

Class Output

- Defined in file_include_miral_miral_output.h

Nested Relationships

Nested Types

- *Struct Output::PhysicalSizeMM*

Class Documentation

class **Output**

Public Types

enum class **Type**

Values:

enumerator **unknown**

enumerator **vga**

enumerator **dvii**

enumerator **dvid**

enumerator **dvia**

enumerator **composite**

enumerator **svideo**

enumerator **lvds**

enumerator **component**

enumerator **ninepindin**

enumerator **displayport**

enumerator **hdmia**

enumerator **hdmib**

enumerator **tv**

enumerator **edp**

Public Functions

explicit **Output**(const mir::graphics::DisplayConfigurationOutput &output)

Output(*Output* const&)

Output &**operator=**(*Output* const&)

~Output()

auto **type**() const -> *Type*
The type of the output.

auto **physical_size_mm**() const -> *PhysicalSizeMM*
The physical size of the output.

auto **connected**() const -> bool
Whether the output is connected.

auto **used**() const -> bool
Whether the output is used in the configuration.

auto **pixel_format**() const -> *MirPixelFormat*
The current output pixel format.

auto **refresh_rate**() const -> double
refresh_rate in Hz

auto **power_mode**() const -> *MirPowerMode*
Current power mode.

auto **orientation**() const -> *MirOrientation*

auto **scale**() const -> float
Requested scale factor for this output, for HiDPI support.

auto **form_factor**() const -> *MirFormFactor*
Form factor of this output; phone display, tablet, monitor, TV, projector...

auto **extents**() const -> Rectangle
The logical rectangle occupied by the output, based on its position, current mode and orientation (rotation)

auto **id**() const -> int
Mir's internal output ID mostly useful for matching against a *miral::WindowInfo::output_id*.

auto **name**() const -> std::string

The output name. This matches that suppled to clients through wl_output.

Remark

Since MirAL 3.8

---

auto **attribute**(std::string const &key) const -> std::optional<std::string>

A custom attribute value.

Remark

Since MirAL 3.8

---

auto **attributes_map**() const -> std::map<std::string const, std::optional<std::string>>

A custom attribute map.

Remark

Since MirAL 3.8

---

auto **valid**() const -> bool

auto **is_same_output**(*Output* const &other) const -> bool

auto **logical_group_id**() const -> int

A positive number if this output is part of a logical output group (aka a display wall) A single display area will stretch across all outputs in a group Zero if this output is not part of a logical group.

struct **PhysicalSizeMM**

Public Members

int **width**

int **height**

Class PrependEventFilter

- Defined in file_include_miral_miral_prepend_event_filter.h

Class Documentation

class **PrependEventFilter**

Public Functions

explicit **PrependEventFilter**(std::function<bool(*MirEvent* const *event)> const &filter)

Prepend an event filter (before any existing filters, including the window manager). The supplied filter should return true if and only if it handles the event as filters later in the list will not be called.

Remark

Since MirAL 3.6

void **operator()**(mir::Server &server)

Template Class SetApplicationAuthorizer

- Defined in file_include_miral_miral_application_authorizer.h

Inheritance Relationships

Base Type

- `public miral::BasicSetApplicationAuthorizer` (*Class BasicSetApplicationAuthorizer*)

Class Documentation

template<typename **Policy**>

class **SetApplicationAuthorizer** : public miral::*BasicSetApplicationAuthorizer*

Public Functions

template<typename ...**Args**>
inline explicit **SetApplicationAuthorizer**(*Args* const&... args)

inline auto **the_custom_application_authorizer**() const -> std::shared_ptr<*Policy*>

Class SetCommandLineHandler

- Defined in file_include_miral_miral_set_command_line_handler.h

Class Documentation

class **SetCommandLineHandler**

Set a handler for any command line options Mir/MirAL does not recognise. This will be invoked if any unrecognised options are found during initialisation. Any unrecognised arguments are passed to this function. The pointers remain valid for the duration of the call only. If set_command_line_handler is not called the default action is to exit by throwing *mir::AbnormalExit* (which will be handled by the exception handler prior to exiting run().

Public Types

using **Handler** = std::function<void(int argc, char const *const *argv)>

Public Functions

explicit **SetCommandLineHandler**(*Handler* const &handler)

~SetCommandLineHandler()

void **operator()**(mir::Server &server) const

Class SetTerminator

- Defined in file_include_miral_miral_set_terminator.h

Class Documentation

class **SetTerminator**

Set handler for termination requests. terminator will be called following receipt of SIGTERM or SIGINT. The default terminator stop()s the server, replacements should probably do the same in addition to any additional shutdown logic.

Public Types

using **Terminator** = std::function<void(int signal)>

Public Functions

explicit **SetTerminator**(*Terminator* const &terminator)

~SetTerminator()

void **operator()**(mir::Server &server) const

Class SetWindowManagementPolicy

- Defined in file_include_miral_miral_set_window_management_policy.h

Class Documentation

class **SetWindowManagementPolicy**

Public Functions

SetWindowManagementPolicy(std::function<std::unique_ptr<*WindowManagementPolicy*>(*WindowManagerTools* const &tools)> const &builder)

~SetWindowManagementPolicy()

void **operator()**(mir::Server &server) const

Class StartupInternalClient

- Defined in file_include_miral_miral_internal_client.h

Class Documentation

class **StartupInternalClient**

Wrapper for running an internal Mir client at startup.

Param client_code
: code implementing the internal client

Param connection_notification
: handler for registering the server-side application

Note: client_code will be executed on its own thread, this must exit

---

Note: connection_notification will be called on a worker thread and must not block

Public Functions

explicit **StartupInternalClient**(std::function<void(struct ::wl_display *display)> client_code,
std::function<void(std::weak_ptr<mir::scene::Session> const session)>
connect_notification)

template<typename **ClientObject**>
inline explicit **StartupInternalClient**(*ClientObject* const &client_object)

~StartupInternalClient()

void **operator()**(mir::Server &server)

Class WaylandExtensions

- Defined in file_include_miral_miral_wayland_extensions.h

Nested Relationships

Nested Types

- *Struct WaylandExtensions::Builder*
- *Class WaylandExtensions::Context*
- *Class WaylandExtensions::EnableInfo*

Class Documentation

class **WaylandExtensions**

Enable configuration of the Wayland extensions enabled at runtime.

This adds the command line options '—wayland-extensions', '—add-wayland-extensions', '—drop-wayland-extensions' and the corresponding MIR_SERVER_* environment variables and config file options.

- The server can add support for additional extensions
- The server can specify the configuration defaults
- Mir's option handling allows the defaults to be overridden

Remark

Since MirAL 2.4

Unnamed Group

static char const *const **zwlr_layer_shell_v1**

Supported wayland extensions that are not enabled by default.

These can be passed into *WaylandExtensions::enable()* to turn them on. Enables shell components such as panels, notifications and lock screens. It is recommended to use this in conjunction with *conditionally_enable()* as malicious clients could potentially use this protocol to steal input focus or otherwise bother the user.

Remark

Since MirAL 2.6

static char const *const **zxdg_output_manager_v1**

Allows clients to retrieve additional information about outputs.

Remark

Since MirAL 2.6

static char const *const **zwlr_foreign_toplevel_manager_v1**

Allows a client to get information and gain control over all toplevels of all clients Useful for taskbars and app switchers Could allow a client to extract information about other programs the user is running.

Remark

Since MirAL 3.1

static char const *const **zwp_virtual_keyboard_manager_v1**

Allows clients to act as a virtual keyboard, useful for on-screen keyboards. Clients are not required to display anything to send keyboard events using this extension, so malicious clients could use it to take actions without user input.

Remark

Since MirAL 3.4

static const char *const **zwp_input_method_v1**

Allows clients (such as on-screen keyboards) to intercept physical key events and act as a source of text input for other clients. Input methods are not required to display anything to use this extension, so malicious clients could use it to intercept keys events or take actions without user input.

Remark

Since MirAL 3.10

static const char *const **zwp_input_panel_v1**

Allows clients to display a surface as an input panel surface. The input panel surface is shown when a text input is active and hidden otherwise. The panel itself can either be attached to the edge of the screen or set to float near the active input. This is often used in conjunction with zwp_input_method_v1.

Remark

Since MirAL 3.10

static char const *const **zwp_input_method_manager_v2**

Allows clients (such as on-screen keyboards) to intercept physical key events and act as a source of text input for other clients. Input methods are not required to display anything to use this extension, so malicious clients could use it to intercept keys events or take actions without user input.

Remark

Since MirAL 3.4

static char const *const **zwlr_screencopy_manager_v1**

Allows clients to take screenshots and record the screen. Only enable for clients that are trusted to view all displayed content, including windows of other apps.

Remark

Since MirAL 3.5

static char const *const **zwlr_virtual_pointer_manager_v1**

Allows clients to act as a virtual pointer, useful for remote control and automation. Clients are not required to display anything to send pointer events using this extension, so malicious clients could use it to take actions without user input.

Remark

Since MirAL 3.6

static char const *const **ext_session_lock_manager_v1**

Allows clients to act as a screen lock.

Remark

Since MirAL 3.6

void **add_extension**(*Builder* const &builder)

Add a bespoke Wayland extension both to "supported" and "enabled by default".

Remark

Since MirAL 2.5

void **add_extension_disabled_by_default**(*Builder* const &builder)

Add a bespoke Wayland extension both to "supported" but not "enabled by default".

Remark

Since MirAL 2.5

auto **enable**(std::string name) -> *WaylandExtensions*&

Enable a Wayland extension by default. The user can still disable it with the drop-wayland-extensions or wayland-extensions options. The extension can be forced to be enabled regardless of user options with *conditionally_enable()*.

Remark

Since MirAL 2.6

auto **disable**(std::string name) -> *WaylandExtensions*&

Disable a Wayland extension by default. The user can still enable it with the add-wayland-extensions or wayland-extensions options. The extension can be forced to be disabled regardless of user options with *conditionally_enable()*.

Remark

Since MirAL 2.6

auto **conditionally_enable**(std::string name, *EnableCallback* const &callback) -> *WaylandExtensions*&

Enable a Wayland extension only when the callback returns true. The callback can use info.user_preference() to respect the extension options the user provided, it is not required. Unlike *enable()* and *disable()*, *conditionally_enable()* can override the user options. The callback may be called multiple times for a each client/extension pair (for example, once each time a client creates or destroys a wl_registry). All client processing will be blocked while the callback is being executed. To minimise the impact on client responsiveness users may want to cache the result of any expensive checks made in the callback.

Remark

Since MirAL 3.4

static auto **recommended**() -> std::set<std::string>

The set of Wayland extensions that Mir recommends. Also the set that is enabled by default upon construction of a *WaylandExtensions* object.

Remark

Since MirAL 2.6

static auto **supported**() -> std::set<std::string>

The set of Wayland extensions that core Mir supports. Does not include bespoke extensions A superset of *recommended()*

Remark

Since MirAL 2.6

Public Types

using **Filter** = std::function<bool(*Application* const &app, char const *protocol)>

Remark

Since MirAL 2.5

using **EnableCallback** = std::function<bool(*EnableInfo* const &info)>

Remark

Since MirAL 3.4

Public Functions

WaylandExtensions()

Default to enabling the extensions recommended by Mir.

void **operator()**(mir::Server &server) const

~WaylandExtensions()

WaylandExtensions(*WaylandExtensions* const&)

auto **operator=**(*WaylandExtensions* const&) -> *WaylandExtensions*&

auto **all_supported**() const -> std::set<std::string>

All Wayland extensions supported. This includes both the *supported()* provided by Mir and any extensions that have been added using *add_extension()*.

Remark

Since MirAL 3.0

struct **Builder**

A *Builder* creates and registers an extension protocol.

Remark

Since MirAL 2.5

Public Members

std::string **name**

Name of the protocol extension's Wayland global.

std::function< std::shared_ptr< void >Context const *context)> build

Functor that creates and registers an extension protocol.

Param context

giving access to:

- the wl_display (so that, for example, the extension can be registered); and,
- allowing server initiated code to be executed on the Wayland mainloop.

Return

a shared pointer to the implementation. (Mir will manage the lifetime)

class **Context**

Context information useful for implementing Wayland extensions.

Remark

Since MirAL 2.5

Public Functions

virtual auto **display**() const -> wl_display* = 0

virtual void **run_on_wayland_mainloop**(std::function<void()> &&work) const = 0

Protected Functions

Context() = default

virtual **~Context**() = default

Context(*Context* const&) = delete

Context &**operator=**(*Context* const&) = delete

class **EnableInfo**

Information that can be used to determine if to enable a conditionally enabled extension.

Remark

Since MirAL 3.4

Public Functions

auto **app**() const -> *Application* const&

The application that is being given access to this extension.

auto **name**() const -> const char*

The name of the extension/global, always the same as given to *conditionally_enable()*

auto **user_preference**() const -> std::optional<bool>

If the user has enabled or disabled this extension one of the wayland extension Mir options.

Class WaylandExtensions::Context

- Defined in file_include_miral_miral_wayland_extensions.h

Nested Relationships

This class is a nested type of *Class WaylandExtensions*.

Class Documentation

class **Context**

Context information useful for implementing Wayland extensions.

Remark

Since MirAL 2.5

Public Functions

virtual auto **display**() const -> wl_display* = 0

virtual void **run_on_wayland_mainloop**(std::function<void()> &&work) const = 0

Protected Functions

Context() = default

virtual **~Context**() = default

Context(*Context* const&) = delete

Context &**operator=**(*Context* const&) = delete

Class WaylandExtensions::EnableInfo

- Defined in file_include_miral_miral_wayland_extensions.h

Nested Relationships

This class is a nested type of *Class WaylandExtensions*.

Class Documentation

class **EnableInfo**

Information that can be used to determine if to enable a conditionally enabled extension.

Remark

Since MirAL 3.4

Public Functions

auto **app**() const -> *Application* const&

The application that is being given access to this extension.

auto **name**() const -> const char*

The name of the extension/global, always the same as given to *conditionally_enable()*

auto **user_preference**() const -> std::optional<bool>

If the user has enabled or disabled this extension one of the wayland extension Mir options.

Class Window

- Defined in file_include_miral_miral_window.h

Class Documentation

class **Window**

Handle class to manage a Mir surface. It may be null (e.g. default initialized)

Unnamed Group

void **resize**(mir::geometry::*Size* const &size)

Not for external use, use *WindowManagerTools::modify_window()* instead.

void **move_to**(mir::geometry::*Point* top_left)

Public Functions

Window()

Window(*Application* const &application, std::shared_ptr<mir::scene::Surface> const &surface)

~Window()

auto **top_left**() const -> mir::geometry::*Point*

The position of the top-left corner of the window frame.

auto **size**() const -> mir::geometry::*Size*

The size of the window frame. Units are logical screen coordinates (not necessarily device pixels). Any decorations are included in the size.

auto **application**() const -> *Application*

The application that created this window.

operator bool() const

Indicates that the *Window* isn't null.

Class WindowManagementPolicy

- Defined in file_include_miral_miral_window_management_policy.h

Inheritance Relationships

Derived Types

- `public TilingWindowManagerPolicy` (*Class TilingWindowManagerPolicy*)
- `public miral::CanonicalWindowManagerPolicy` (*Class CanonicalWindowManagerPolicy*)
- `public miral::MinimalWindowManager` (*Class MinimalWindowManager*)

Class Documentation

class **WindowManagementPolicy**

The interface through which the window management policy is determined.

Subclassed by *TilingWindowManagerPolicy*, *miral::CanonicalWindowManagerPolicy*, *miral::MinimalWindowManager*

handle events originating from the client

The policy is expected to update the model as appropriate

virtual void **handle_window_ready**(*WindowInfo* &window_info) = 0

notification that the first buffer has been posted

Parameters

window_info – the window

virtual void **handle_modify_window**(*WindowInfo* &window_info, *WindowSpecification* const &modifications) = 0

request from client to modify the window specification.

Parameters

- **window_info** – the window
- **modifications** – the requested changes

Note: the request has already been validated against the type definition

---

virtual void **handle_raise_window**(*WindowInfo* &window_info) = 0

request from client to raise the window

Parameters

window_info – the window

Note: the request has already been validated against the requesting event

---

virtual auto **confirm_placement_on_display**(*WindowInfo* const &window_info, *MirWindowState* new_state, Rectangle const &new_placement) -> Rectangle = 0

Confirm (and optionally adjust) the placement of a window on the display.

Called when (re)placing fullscreen, maximized, horizontally maximised and vertically maximized windows to allow adjustment for decorations.

Parameters

- **window_info** – the window
- **new_state** – the new state
- **new_placement** – the suggested placement

Returns

the confirmed placement of the window

handle events originating from user

The policy is expected to interpret (and optionally consume) the event

virtual bool **handle_keyboard_event**(MirKeyboardEvent const *event) = 0

keyboard event handler

Parameters

event – the event

Returns

whether the policy has consumed the event

virtual bool **handle_touch_event**(MirTouchEvent const *event) = 0

touch event handler

Parameters

event – the event

Returns

whether the policy has consumed the event

virtual bool **handle_pointer_event**(MirPointerEvent const *event) = 0

pointer event handler

Parameters

event – the event

Returns

whether the policy has consumed the event

notification of WM events that the policy may need to track.

virtual void **advise_new_app**(*ApplicationInfo* &application)

Notification that a new application has connected.

Parameters

application – the application

virtual void **advise_delete_app**(*ApplicationInfo* const &application)

Notification that an application has disconnected.

Parameters

application – the application

virtual void **advise_new_window**(*WindowInfo* const &window_info)

Notification that a window has been created.

Parameters

window_info – the window

virtual void **advise_focus_lost**(*WindowInfo* const &window_info)

Notification that a window has lost focus.

Parameters

window_info – the window

virtual void **advise_focus_gained**(*WindowInfo* const &window_info)

Notification that a window has gained focus.

Parameters

window_info – the window

virtual void **advise_state_change**(*WindowInfo* const &window_info, *MirWindowState* state)

Notification that a window is about to change state.

Parameters

- **window_info** – the window
- **state** – the new state

virtual void **advise_move_to**(*WindowInfo* const &window_info, Point top_left)

Notification that a window is about to move.

Parameters

- **window_info** – the window
- **top_left** – the new position

virtual void **advise_resize**(*WindowInfo* const &window_info, Size const &new_size)

Notification that a window is about to resize.

Parameters

- **window_info** – the window
- **new_size** – the new size

virtual void **advise_delete_window**(*WindowInfo* const &window_info)

Notification that a window is about to be destroyed.

Parameters

window_info – the window

virtual void **advise_raise**(std::vector<*Window*> const &windows)

Notification that windows are being raised to the top.

These windows are ordered with parents before children, and form a single tree rooted at the first element.

Parameters

windows – the windows

Note: The relative Z-order of these windows will be maintained, they will be raised en bloc.

virtual void **advise_adding_to_workspace**(std::shared_ptr<Workspace> const &workspace, std::vector<*Window*> const &windows)

Notification that windows are being added to a workspace.

These windows are ordered with parents before children, and form a single tree rooted at the first element.

Parameters

- **workspace** – the workspace
- **windows** – the windows

virtual void **advise_removing_from_workspace**(std::shared_ptr<Workspace> const &workspace, std::vector<*Window*> const &windows)

Notification that windows are being removed from a workspace.

These windows are ordered with parents before children, and form a single tree rooted at the first element.

Parameters

- **workspace** – the workspace
- **windows** – the windows

handle requests originating from the client

The policy is expected to update the model as appropriate

virtual void **handle_request_move**(*WindowInfo* &window_info, MirInputEvent const *input_event) = 0

request from client to initiate move

Parameters

- **window_info** – the window
- **input_event** – the requesting event

Note: the request has already been validated against the requesting event

virtual void **handle_request_resize**(*WindowInfo* &window_info, MirInputEvent const *input_event, *MirResizeEdge* edge) = 0

request from client to initiate resize

Parameters

- **window_info** – the window
- **input_event** – the requesting event
- **edge** – the edge(s) being dragged

Note: the request has already been validated against the requesting event

notification of changes to the (connected, active) outputs.

virtual void **advise_output_create**(*Output* const &output)

virtual void **advise_output_update**(*Output* const &updated, *Output* const &original)

virtual void **advise_output_delete**(*Output* const &output)

notification of changes to the current application zones

An application zone is the area a maximized application will fill.

There is often (but not necessarily) one zone per output. The areas normal applications windows should avoid (such as the areas covered by panels) will not be part of an application zone

virtual void **advise_application_zone_create**(*Zone* const &application_zone)

virtual void **advise_application_zone_update**(*Zone* const &updated, *Zone* const &original)

virtual void **advise_application_zone_delete**(*Zone* const &application_zone)

Public Functions

virtual void **advise_begin**()

before any related calls begin

virtual void **advise_end**()

after any related calls end

virtual auto **place_new_window**(*ApplicationInfo* const &app_info, *WindowSpecification* const &requested_specification) -> *WindowSpecification* = 0

Customize initial window placement.

Parameters

- **app_info** – the application requesting a new window
- **requested_specification** – the requested specification (updated with default placement)

Returns

the customized specification

virtual auto **confirm_inherited_move**(*WindowInfo* const &window_info, Displacement movement) -> Rectangle = 0

Confirm (and optionally adjust) the motion of a child window when the parent is moved.

Parameters

- **window_info** – the window
- **movement** – the movement of the parent

Returns

the confirmed placement of the window

virtual **~WindowManagementPolicy**() = default

WindowManagementPolicy() = default

WindowManagementPolicy(*WindowManagementPolicy* const&) = delete

WindowManagementPolicy &**operator=**(*WindowManagementPolicy* const&) = delete

Class WindowManagerOptions

- Defined in file_include_miral_miral_window_management_options.h

Class Documentation

class **WindowManagerOptions**

Public Functions

WindowManagerOptions() = delete

inline explicit **WindowManagerOptions**(std::initializer_list<*WindowManagerOption*> const &policies)

void **operator()**(mir::Server &server) const

Public Members

std::vector<*WindowManagerOption*> const **policies**

Class WindowManagerTools

- Defined in file_include_miral_miral_window_manager_tools.h

Class Documentation

class **WindowManagerTools**

Window management functions for querying and updating MirAL's model.

Query & Update Model

These functions assume that the BasicWindowManager data structures can be accessed freely.

I.e. they should only be used by a thread that has called the *WindowManagementPolicy* methods (where any necessary locks are held) or via a *invoke_under_lock()* callback.

auto **count_applications**() const -> unsigned int

count the applications

Returns

number of applications

void **for_each_application**(std::function<void(*ApplicationInfo* &info)> const &functor)

execute functor for each application

Parameters

functor – the functor

auto **find_application**(std::function<bool(*ApplicationInfo* const &info)> const &predicate) -> *Application*

find an application meeting the predicate

Parameters

predicate – the predicate

Returns

the application

auto **info_for**(std::weak_ptr<mir::scene::Session> const &session) const -> *ApplicationInfo*&

retrieve metadata for an application

Parameters

session – the application session

Returns

the metadata

auto **info_for**(std::weak_ptr<mir::scene::Surface> const &surface) const -> *WindowInfo*&

retrieve metadata for a window

Parameters

surface – the window surface

Returns

the metadata

auto **info_for**(*Window* const &window) const -> *WindowInfo*&

retrieve metadata for a window

Parameters

window – the window

Returns

the metadata

auto **info_for_window_id**(std::string const &id) const -> *WindowInfo*&

retrieve metadata for a persistent surface id

Parameters

id – the persistent surface id

Throws
`invalid_argument` – or runtime_error if the id is badly formatted/doesn't identify a current window
Returns
the metadata

auto **id_for_window**(*Window* const &window) const -> std::string

retrieve the persistent surface id for a window
Parameters
window – the window
Returns
the persistent surface id

void **ask_client_to_close**(*Window* const &window)

Send close request to the window.

auto **active_window**() const -> *Window*

retrieve the active window

auto **select_active_window**(*Window* const &hint) -> *Window*

select a new active window based on the hint
Parameters
hint – the hint
Returns
the new active window

void **drag_active_window**(mir::geometry::*Displacement* movement)

move the active window

void **drag_window**(*Window* const &window, mir::geometry::*Displacement* movement)

move the window

void **focus_next_application**()

make the next application active

void **focus_prev_application**()

make the previous application active

Remark

Since MirAL 2.5

---

void **focus_next_within_application**()

make the next surface active within the active application

void **focus_prev_within_application**()

make the prev surface active within the active application

auto **window_to_select_application**(const *Application*) const -> std::optional<*Window*>

If possible, returns the application window to select, otherwise `std::nullopt`

Remark

Since MirAL 3.10

auto **window_at**(mir::geometry::*Point* cursor) const -> *Window*

Find the topmost window at the cursor.

auto **active_output**() -> mir::geometry::*Rectangle* const

Find the active output area.

auto **active_application_zone**() const -> *Zone*

Find the active zone area.

Remark

Since MirAL 3.0

void **raise_tree**(*Window* const &root)

Raise window and all its children.

void **swap_tree_order**(*Window* const &first, *Window* const &second)

Swaps the position of the windows in regards to Z order.

Remark

SinceMirAL 3.10

Parameters

- **first** –
- **second** –

void **send_tree_to_back**(*Window* const &root)

Moves the window to the bottom of the Z order remark Since MirAL 3.10.

void **modify_window**(*WindowInfo* &window_info, *WindowSpecification* const &modifications)

Apply modifications to a window.

void **modify_window**(*Window* const &window, *WindowSpecification* const &modifications)

Apply modifications to a window.

void **place_and_size_for_state**(*WindowSpecification* &modifications, *WindowInfo* const &window_info) const

Set a default size and position to reflect state change.

auto **create_workspace**() -> std::shared_ptr<Workspace>

Create a workspace.

Remark

the tools hold only a weak_ptr<> to the workspace - there is no need for an explicit “destroy”.

Returns
a shared_ptr owning the workspace

void **add_tree_to_workspace**(*Window* const &window, std::shared_ptr<Workspace> const &workspace)

Add the tree containing window to a workspace.

Parameters

- **window** – the window
- **workspace** – the workspace;

void **remove_tree_from_workspace**(*Window* const &window, std::shared_ptr<Workspace> const &workspace)

Remove the tree containing window from a workspace.

Parameters

- **window** – the window
- **workspace** – the workspace;

void **move_workspace_content_to_workspace**(std::shared_ptr<Workspace> const &to_workspace, std::shared_ptr<Workspace> const &from_workspace)

Moves all the content from one workspace to another.

Parameters

- **from_workspace** – the workspace to move the windows from;
- **to_workspace** – the workspace to move the windows to;

void **for_each_workspace_containing**(*Window* const &window, std::function<void(std::shared_ptr<Workspace> const &workspace)> const &callback)

invoke callback with each workspace containing window

Parameters

- **window** –
- **callback** –

> **Warning:** it is unsafe to add or remove windows from workspaces from the callback during enumeration

void **for_each_window_in_workspace**(std::shared_ptr<Workspace> const &workspace, std::function<void(*Window* const &window)> const &callback)

invoke callback with each window contained in workspace

Parameters

- **workspace** –
- **callback** –

> **Warning:** it is unsafe to add or remove windows from workspaces from the callback during enumeration

Public Functions

explicit **WindowManagerTools**(WindowManagerToolsImplementation *tools)

WindowManagerTools(*WindowManagerTools* const&)

WindowManagerTools &**operator=**(*WindowManagerTools* const&)

~WindowManagerTools()

void **invoke_under_lock**(std::function<void()> const &callback)

Multi-thread support Allows threads that don't hold a lock on the model to acquire one and call the "Update Model" member functions.

This should NOT be used by a thread that has called the *WindowManagementPolicy* methods (and already holds the lock).

Class WindowSpecification

- Defined in file_include_miral_miral_window_specification.h

Nested Relationships

Nested Types

- *Struct WindowSpecification::AspectRatio*

Class Documentation

class **WindowSpecification**

Unnamed Group

auto **min_width**() -> mir::*optional_value*<Width>&

Constrains how a window can be resized, see the corresponding properties on *WindowInfo* for details.

auto **min_height**() -> mir::*optional_value*<Height>&

auto **max_width**() -> mir::*optional_value*<Width>&

auto **max_height**() -> mir::*optional_value*<Height>&

auto **width_inc**() -> mir::*optional_value*<DeltaX>&

auto **height_inc**() -> mir::*optional_value*<DeltaY>&

auto **min_aspect**() -> mir::*optional_value*<*AspectRatio*>&

auto **max_aspect**() -> mir::*optional_value*<*AspectRatio*>&

Unnamed Group

auto **depth_layer**() const -> mir::*optional_value*<*MirDepthLayer*> const&

The depth layer of a child window is updated with the depth layer of its parent, but can be overridden.

auto **depth_layer**() -> mir::*optional_value*<*MirDepthLayer*>&

Unnamed Group

auto **attached_edges**() const -> mir::*optional_value*<*MirPlacementGravity*> const&

The set of window eges that are attched to edges of the output If attached to perpendicular edges, it is attached to the corner where the two edges intersect If attached to oposite edges (eg left and right), it is stretched across the output in that direction If all edges are specified, it takes up the entire output.

auto **attached_edges**() -> mir::*optional_value*<*MirPlacementGravity*>&

Unnamed Group

auto **exclusive_rect**() const -> mir::*optional_value*<mir::*optional_value*<mir::geometry::*Rectangle*>> const&

The area over which the window should not be occluded Only meaningful for windows attached to an edge If the outer optional is unset (the default), the window's exclusive rect is not changed by this spec If the outer optional is set but the inner is not, the window's exclusive rect is cleared.

auto **exclusive_rect**() -> mir::*optional_value*<mir::*optional_value*<mir::geometry::*Rectangle*>>&

Unnamed Group

auto **application_id**() const -> mir::*optional_value*<std::string> const&

The D-bus service name and basename of the app's .desktop file See http://standards.freedesktop.org/desktop-entry-spec/.

auto **application_id**() -> mir::*optional_value*<std::string>&

Unnamed Group

auto **server_side_decorated**() const -> mir::*optional_value*<bool> const&

If this window should have server-side decorations provided by Mir Currently, Mir only respects this value during surface construction.

auto **server_side_decorated**() -> mir::*optional_value*<bool>&

Unnamed Group

auto **focus_mode**() const -> mir::*optional_value*<*MirFocusMode*> const&

How the window should gain and lose focus.

Remark

Since MirAL 3.3

---

auto **focus_mode**() -> mir::*optional_value*<*MirFocusMode*>&

Unnamed Group

auto **visible_on_lock_screen**() const -> mir::*optional_value*<bool> const&

If this surface should be shown while the compositor is locked.

Remark

Since MirAL 3.9

---

auto **visible_on_lock_screen**() -> mir::*optional_value*<bool>&

Public Types

enum class **InputReceptionMode**

Values:

enumerator **normal**

enumerator **receives_all_input**

Public Functions

WindowSpecification()

WindowSpecification(*WindowSpecification* const &that)

auto **operator=**(*WindowSpecification* const &that) -> *WindowSpecification*&

WindowSpecification(mir::shell::SurfaceSpecification const &spec)

~WindowSpecification()

auto **top_left**() const -> mir::*optional_value*<Point> const&

auto **size**() const -> mir::*optional_value*<Size> const&

auto **name**() const -> mir::*optional_value*<std::string> const&

auto **output_id**() const -> mir::*optional_value*<int> const&

auto **type**() const -> mir::*optional_value*<*MirWindowType*> const&

auto **state**() const -> mir::*optional_value*<*MirWindowState*> const&

auto **preferred_orientation**() const -> mir::*optional_value*<*MirOrientationMode*> const&

auto **aux_rect**() const -> mir::*optional_value*<Rectangle> const&

auto **placement_hints**() const -> mir::*optional_value*<*MirPlacementHints*> const&

auto **window_placement_gravity**() const -> mir::*optional_value*<*MirPlacementGravity*> const&

auto **aux_rect_placement_gravity**() const -> mir::*optional_value*<*MirPlacementGravity*> const&

auto **aux_rect_placement_offset**() const -> mir::*optional_value*<Displacement> const&

auto **min_width**() const -> mir::*optional_value*<Width> const&

auto **min_height**() const -> mir::*optional_value*<Height> const&

auto **max_width**() const -> mir::*optional_value*<Width> const&

auto **max_height**() const -> mir::*optional_value*<Height> const&

auto **width_inc**() const -> mir::*optional_value*<DeltaX> const&

auto **height_inc**() const -> mir::*optional_value*<DeltaY> const&

auto **min_aspect**() const -> mir::*optional_value*<*AspectRatio*> const&

auto **max_aspect**() const -> mir::*optional_value*<*AspectRatio*> const&

auto **parent**() const -> mir::*optional_value*<std::weak_ptr<mir::scene::Surface>> const&

auto **input_shape**() const -> mir::*optional_value*<std::vector<Rectangle>> const&

auto **input_mode**() const -> mir::*optional_value*<*InputReceptionMode*> const&

auto **shell_chrome**() const -> mir::*optional_value*<*MirShellChrome*> const&

auto **confine_pointer**() const -> mir::*optional_value*<*MirPointerConfinementState*> const&

auto **userdata**() const -> mir::*optional_value*<std::shared_ptr<void>> const&

auto **top_left**() -> mir::*optional_value*<Point>&

auto **size**() -> mir::*optional_value*<Size>&

The new size of the window frame (including any decorations). Will be adjusted based on *min_width()*, *WindowInfo::max_width()*, *WindowInfo::min_height()*, *WindowInfo::max_height()*, *WindowInfo::min_aspect()*, *WindowInfo::max_aspect()*, *WindowInfo::width_inc()* and *WindowInfo::height_inc()*. Set these to properties to their default values if they should be ignored.

auto **name**() -> mir::*optional_value*<std::string>&

auto **output_id**() -> mir::*optional_value*<int>&

auto **type**() -> mir::*optional_value*<*MirWindowType*>&

auto **state**() -> mir::*optional_value*<*MirWindowState*>&

auto **preferred_orientation**() -> mir::*optional_value*<*MirOrientationMode*>&

auto **aux_rect**() -> mir::*optional_value*<Rectangle>&

Relative to window's surface's CONTENT offset and size (not equal to the top_left and size exposed by this interface if server-side decorations are in use)

auto **placement_hints**() -> mir::*optional_value*<*MirPlacementHints*>&

auto **window_placement_gravity**() -> mir::*optional_value*<*MirPlacementGravity*>&

auto **aux_rect_placement_gravity**() -> mir::*optional_value*<*MirPlacementGravity*>&

auto **aux_rect_placement_offset**() -> mir::*optional_value*<Displacement>&

auto **parent**() -> mir::*optional_value*<std::weak_ptr<mir::scene::Surface>>&

auto **input_shape**() -> mir::*optional_value*<std::vector<Rectangle>>&

auto **input_mode**() -> mir::*optional_value*<*InputReceptionMode*>&

auto **shell_chrome**() -> mir::*optional_value*<*MirShellChrome*>&

auto **confine_pointer**() -> mir::*optional_value*<*MirPointerConfinementState*>&

auto **userdata**() -> mir::*optional_value*<std::shared_ptr<void>>&

struct **AspectRatio**

Public Members

unsigned **width**

unsigned **height**

Class X11Support

- Defined in file_include_miral_miral_x11_support.h

Class Documentation

class **X11Support**

Add user configuration options for X11 support.

Remark

Since MirAL 2.4

Note: From MirAL 4.1, the `enable-x11` option requires an argument (e.g., `enable-x11=true` to enable X11 support).

Public Functions

void **operator()**(mir::Server &server) const

X11Support()

~X11Support()

X11Support(*X11Support* const&)

auto **operator=**(*X11Support* const&) -> *X11Support*&

auto **default_to_enabled**() -> *X11Support*&

Set X11 support as enabled by default.

Remark

Since MirAL 4.1

Returns

A reference to the modified *miral::X11Support* object with the updated default configuration.

Class Zone

- Defined in file_include_miral_miral_zone.h

Class Documentation

class **Zone**

A rectangular area of the display. Not tied to a specific output.

Public Functions

Zone(Rectangle const &extents)

Create a new zone with the given extents.

Zone(*Zone* const &other)

Makes a copy of the underlying private data.

Zone &**operator=**(*Zone* const &other)

Copies private data by value.

~Zone()

auto **operator==**(*Zone* const &other) const -> bool

Returns true only if all properties including IDs match.

auto **is_same_zone**(*Zone* const &other) const -> bool

Returns if true if zone IDs match, even if extents are different.

auto **extents**() const -> Rectangle

The area of this zone in global display coordinates.

void **extents**(Rectangle const &extents)

Set the extents of this zone Does not make this a different zone.

auto **id**() const -> int

An arbitrary number that uniquely identifies this zone, reguardless of how it is resized and moved.

Remark

Since MirAL 3.6

Class MirEglSurface

- Defined in file_examples_miral-shell_spinner_miregl.h

Inheritance Relationships

Base Type

- `private WaylandSurface` (*Class WaylandSurface*)

Class Documentation

class **MirEglSurface** : private *WaylandSurface*

Public Functions

MirEglSurface(std::shared_ptr<MirEglApp> const &mir_egl_app, struct *wl_output* *wl_output)

~MirEglSurface()

template<typename **Painter**>
inline void **paint**(*Painter* const &functor)

Class Compositor

- Defined in file_include_miroil_miroil_compositor.h

Class Documentation

class **Compositor**

Public Functions

virtual **~Compositor**()

Compositor &**operator=**(*Compositor* const&) = delete

virtual void **start**() = 0

virtual void **stop**() = 0

Protected Functions

Compositor() = default

Compositor(*Compositor* const&) = delete

Class DisplayConfigurationControllerWrapper

- Defined in file_include_miroil_miroil_display_configuration_controller_wrapper.h

Class Documentation

class **DisplayConfigurationControllerWrapper**

Public Functions

DisplayConfigurationControllerWrapper(std::shared_ptr<mir::shell::DisplayConfigurationController> const &wrapped)

~DisplayConfigurationControllerWrapper() = default

void **set_base_configuration**(std::shared_ptr<mir::graphics::DisplayConfiguration> const &conf)

Set the base display configuration.

This is the display configuration that is used by default, but will be overridden by a client's requested configuration if that client is focused.

Parameters

conf – **[in]** The new display configuration to set

Class DisplayConfigurationPolicy

- Defined in file_include_miroil_miroil_display_configuration_policy.h

Class Documentation

class **DisplayConfigurationPolicy**

Public Functions

DisplayConfigurationPolicy()

virtual **~DisplayConfigurationPolicy**()

DisplayConfigurationPolicy(*DisplayConfigurationPolicy* const&) = delete

DisplayConfigurationPolicy &**operator=**(*DisplayConfigurationPolicy* const&) = delete

virtual void **apply_to**(mir::graphics::DisplayConfiguration &conf) = 0

Class DisplayConfigurationStorage

- Defined in file_include_miroil_miroil_display_configuration_storage.h

Class Documentation

class **DisplayConfigurationStorage**

Public Functions

virtual **~DisplayConfigurationStorage**() = default

virtual void **save**(const *DisplayId*&, const *DisplayConfigurationOptions*&) = 0

virtual bool **load**(const *DisplayId*&, *DisplayConfigurationOptions*&) const = 0

Class DisplayListenerWrapper

- Defined in file_include_miroil_miroil_display_listener_wrapper.h

Class Documentation

class **DisplayListenerWrapper**

Public Functions

DisplayListenerWrapper(std::shared_ptr<mir::compositor::DisplayListener> const &display_listener)

~DisplayListenerWrapper()

void **add_display**(mir::geometry::*Rectangle* const &area)

void **remove_display**(mir::geometry::*Rectangle* const &area)

Class EventBuilder

- Defined in file_include_miroil_miroil_event_builder.h

Nested Relationships

Nested Types

- *Class EventBuilder::EventInfo*

Class Documentation

class **EventBuilder**

Public Functions

EventBuilder()

virtual **~EventBuilder**()

void **add_touch**(*MirEvent* &event, *MirTouchId* touch_id, *MirTouchAction* action, *MirTouchTooltype* tooltype, float x_axis_value, float y_axis_value, float pressure_value, float touch_major_value, float touch_minor_value, float size_value)

mir::*EventUPtr* **make_key_event**(*MirInputDeviceId* device_id, std::chrono::nanoseconds timestamp, std::vector<uint8_t> const &cookie, *MirKeyboardAction* action, xkb_keysym_t keysym, int scan_code, *MirInputEventModifiers* modifiers)

mir::*EventUPtr* **make_touch_event**(*MirInputDeviceId* device_id, std::chrono::nanoseconds timestamp, std::vector<uint8_t> const &mac, *MirInputEventModifiers* modifiers)

mir::*EventUPtr* **make_pointer_event**(*MirInputDeviceId* device_id, std::chrono::nanoseconds timestamp, std::vector<uint8_t> const &mac, *MirInputEventModifiers* modifiers, *MirPointerAction* action, *MirPointerButtons* buttons_pressed, float x_axis_value, float y_axis_value, float hscroll_value, float vscroll_value, float relative_x_value, float relative_y_value)

EventInfo ***find_info**(ulong qtTimestamp)

void **store**(const MirInputEvent *mirInputEvent, ulong qtTimestamp)

class **EventInfo**

Public Functions

void **store**(const MirInputEvent *mirInputEvent, ulong qtTimestamp)

Public Members

ulong **timestamp**

MirInputDeviceId **device_id**

std::vector<uint8_t> **cookie**

float **relative_x** = {0}

float **relative_y** = {0}

Class EventBuilder::EventInfo

- Defined in file_include_miroil_miroil_event_builder.h

Nested Relationships

This class is a nested type of *Class EventBuilder*.

Class Documentation

class **EventInfo**

Public Functions

void **store**(const MirInputEvent *mirInputEvent, ulong qtTimestamp)

Public Members

ulong **timestamp**

MirInputDeviceId **device_id**

std::vector<uint8_t> **cookie**

float **relative_x** = {0}

float **relative_y** = {0}

Class GLBuffer

- Defined in file_include_miroil_miroil_mirbuffer.h

Class Documentation

class **GLBuffer**

Public Functions

~GLBuffer()

explicit **GLBuffer**(std::shared_ptr<mir::graphics::Buffer> const &buffer)

bool **has_alpha_channel**() const

mir::geometry::*Size* **size**() const

void **bind**()

void **reset**(std::shared_ptr<mir::graphics::Buffer> const &buffer)

void **reset**()

bool **empty**()

Public Static Functions

static std::shared_ptr<*GLBuffer*> **from_mir_buffer**(std::shared_ptr<mir::graphics::Buffer> const &buffer)

Class InputDevice

- Defined in file_include_miroil_miroil_input_device.h

Class Documentation

class **InputDevice**

Public Functions

InputDevice(std::shared_ptr<mir::input::Device> const &device)

InputDevice(*InputDevice* const &src)

InputDevice(*InputDevice* &&src)

InputDevice()

~InputDevice()

auto **operator=**(*InputDevice* const &src) -> *InputDevice*&

auto **operator=**(*InputDevice* &&src) -> *InputDevice*&

bool **operator==**(*InputDevice* const &other)

void **apply_keymap**(std::string const &layout, std::string const &variant)

auto **get_device_id**() -> *MirInputDeviceId*

auto **get_device_name**() -> std::string

auto **is_keyboard**() -> bool

auto **is_alpha_numeric**() -> bool

Class InputDeviceObserver

- Defined in file_include_miroil_miroil_input_device_observer.h

Class Documentation

class **InputDeviceObserver**

Public Functions

InputDeviceObserver() = default

InputDeviceObserver(*InputDeviceObserver* const&) = delete

InputDeviceObserver &**operator=**(*InputDeviceObserver* const&) = delete

virtual **~InputDeviceObserver**()

virtual void **device_added**(miroil::*InputDevice* device) = 0

virtual void **device_removed**(miroil::*InputDevice* device) = 0

Class MirPromptSession

- Defined in file_include_miroil_miroil_mir_prompt_session.h

Class Documentation

class **MirPromptSession**

Public Functions

MirPromptSession(::*MirPromptSession* *promptSession)

MirPromptSession(*MirPromptSession* const &src)

MirPromptSession(*MirPromptSession* &&src)

~MirPromptSession()

auto **operator=**(*MirPromptSession* const &src) -> *MirPromptSession*&

auto **operator=**(*MirPromptSession* &&src) -> *MirPromptSession*&

bool **operator==**(*MirPromptSession* const &other)

bool **new_fds_for_prompt_providers**(unsigned int no_of_fds, *MirClientFdCallback* callback, void *context)

Public Members

::*MirPromptSession* ***prompt_session**

Class MirServerHooks

- Defined in file_include_miroil_miroil_mir_server_hooks.h

Class Documentation

class **MirServerHooks**

Public Functions

MirServerHooks()

void **operator()**(mir::Server &server)

auto **the_prompt_session_listener**() const -> *PromptSessionListener**

auto **the_prompt_session_manager**() const -> std::shared_ptr<mir::scene::PromptSessionManager>

auto **the_mir_display**() const -> std::shared_ptr<mir::graphics::Display>

auto **the_display_configuration_controller**() const -> std::shared_ptr<mir::shell::DisplayConfigurationController>

void **create_named_cursor**(*CreateNamedCursor* func)

void **create_input_device_observer**(std::shared_ptr<*InputDeviceObserver*> &observer)

void **create_prompt_session_listener**(std::shared_ptr<*PromptSessionListener*> listener)

Class OpenGLContext

- Defined in file_include_miroil_miroil_open_gl_context.h

Class Documentation

class **OpenGLContext**

Public Functions

OpenGLContext(mir::graphics::GLConfig *gl_config)

void **operator()**(mir::Server &server)

auto **the_open_gl_config**() const -> std::shared_ptr<mir::graphics::GLConfig>

Class PersistDisplayConfig

- Defined in file_include_miroil_miroil_persist_display_config.h

Class Documentation

class **PersistDisplayConfig**

Restores the saved display configuration and saves changes to the base configuration.

Public Types

using **DisplayConfigurationPolicyWrapper** = std::function<std::shared_ptr<*DisplayConfigurationPolicy*>(std::shared_ptr<mir::graphics::DisplayConfigurationPolicy> const &wrapped)>

Public Functions

~PersistDisplayConfig()

PersistDisplayConfig(*PersistDisplayConfig* const&)

auto **operator=**(*PersistDisplayConfig* const&) -> *PersistDisplayConfig*&

PersistDisplayConfig(std::shared_ptr<*DisplayConfigurationStorage*> const &storage,
DisplayConfigurationPolicyWrapper const &custom_wrapper)

void **operator()**(mir::Server &server)

Class PromptSessionListener

- Defined in file_include_miroil_miroil_prompt_session_listener.h

Class Documentation

class **PromptSessionListener**

Public Functions

virtual **~PromptSessionListener**()

PromptSessionListener &**operator=**(*PromptSessionListener* const&) = delete

virtual void **starting**(std::shared_ptr<mir::scene::PromptSession> const &prompt_session) = 0

virtual void **stopping**(std::shared_ptr<mir::scene::PromptSession> const &prompt_session) = 0

virtual void **suspending**(std::shared_ptr<mir::scene::PromptSession> const &prompt_session) = 0

virtual void **resuming**(std::shared_ptr<mir::scene::PromptSession> const &prompt_session) = 0

virtual void **prompt_provider_added**(mir::scene::PromptSession const &prompt_session,
std::shared_ptr<mir::scene::Session> const &prompt_provider) = 0

virtual void **prompt_provider_removed**(mir::scene::PromptSession const &prompt_session,
std::shared_ptr<mir::scene::Session> const &prompt_provider) = 0

Protected Functions

PromptSessionListener() = default

PromptSessionListener(*PromptSessionListener* const&) = delete

Class PromptSessionManager

- Defined in file_include_miroil_miroil_prompt_session_manager.h

Class Documentation

class **PromptSessionManager**

Public Functions

PromptSessionManager(std::shared_ptr<mir::scene::PromptSessionManager> const &prompt_session_manager)

PromptSessionManager(*PromptSessionManager* const &src)

PromptSessionManager(*PromptSessionManager* &&src)

~PromptSessionManager()

auto **operator=**(*PromptSessionManager* const &src) -> *PromptSessionManager*&

auto **operator=**(*PromptSessionManager* &&src) -> *PromptSessionManager*&

bool **operator==**(*PromptSessionManager* const &other)

auto **application_for**(std::shared_ptr<mir::scene::PromptSession> const &prompt_session) const -> miral::*Application*

void **resume_prompt_session**(std::shared_ptr<mir::scene::PromptSession> const &prompt_session) const

void **stop_prompt_session**(std::shared_ptr<mir::scene::PromptSession> const &prompt_session) const

void **suspend_prompt_session**(std::shared_ptr<mir::scene::PromptSession> const &prompt_session) const

Class SetCompositor

- Defined in file_include_miroil_miroil_set_compositor.h

Class Documentation

class **SetCompositor**

Public Functions

SetCompositor(ConstructorFunction constructor, InitFunction init)

void **operator()**(mir::Server &server)

Class Surface

- Defined in file_include_miroil_miroil_surface.h

Class Documentation

class **Surface**

Public Functions

Surface(std::shared_ptr<mir::scene::Surface> wrapped)

~Surface() = default

mir::scene::Surface ***get_wrapped**() const

void **add_observer**(std::shared_ptr<miroil::*SurfaceObserver*> const &observer)

void **remove_observer**(std::shared_ptr<miroil::*SurfaceObserver*> const &observer)

int **buffers_ready_for_compositor**(void const *compositor_id) const

mir::graphics::RenderableList **generate_renderables**(miroil::*CompositorID* id) const

bool **is_confined_to_window**()

void **set_orientation**(*MirOrientation* orientation)

void **set_confine_pointer_state**(*MirPointerConfinementState* state)

std::shared_ptr<mir::scene::Surface> **parent**() const

mir::geometry::*Point* **top_left**() const
Top-left corner (of the window frame if present)

bool **visible**() const

int **configure**(*MirWindowAttrib* attrib, int value)

int **query**(*MirWindowAttrib* attrib) const

void **set_keymap**(*MirInputDeviceId* id, std::string const &model, std::string const &layout, std::string const &variant, std::string const &options)

Class SurfaceObserver

- Defined in file_include_miroil_miroil_surface_observer.h

Class Documentation

class **SurfaceObserver**

Public Functions

SurfaceObserver() = default

SurfaceObserver(*SurfaceObserver* const&) = delete

SurfaceObserver &**operator=**(*SurfaceObserver* const&) = delete

virtual **~SurfaceObserver**() = default

virtual void **attrib_changed**(mir::scene::Surface const *surf, *MirWindowAttrib* attrib, int value) = 0

virtual void **window_resized_to**(mir::scene::Surface const *surf, mir::geometry::*Size* const &window_size) = 0

virtual void **content_resized_to**(mir::scene::Surface const *surf, mir::geometry::*Size* const &content_size) = 0

virtual void **moved_to**(mir::scene::Surface const *surf, mir::geometry::*Point* const &top_left) = 0

virtual void **hidden_set_to**(mir::scene::Surface const *surf, bool hide) = 0

virtual void **frame_posted**(mir::scene::Surface const *surf, int frames_available, mir::geometry::*Size* const &size) = 0

virtual void **alpha_set_to**(mir::scene::Surface const *surf, float alpha) = 0

virtual void **orientation_set_to**(mir::scene::Surface const *surf, *MirOrientation* orientation) = 0

virtual void **transformation_set_to**(mir::scene::Surface const *surf, glm::mat4 const &t) = 0

virtual void **cursor_image_set_to**(mir::scene::Surface const *surf, mir::graphics::CursorImage const &image) = 0

virtual void **client_surface_close_requested**(mir::scene::Surface const *surf) = 0

virtual void **keymap_changed**(mir::scene::Surface const *surf, *MirInputDeviceId* id, std::string const &model, std::string const &layout, std::string const &variant, std::string const &options) = 0

virtual void **renamed**(mir::scene::Surface const *surf, char const *name) = 0

virtual void **cursor_image_removed**(mir::scene::Surface const *surf) = 0

virtual void **placed_relative**(mir::scene::Surface const *surf, mir::geometry::*Rectangle* const &placement) = 0

virtual void **input_consumed**(mir::scene::Surface const *surf, *MirEvent* const *event) = 0

virtual void **start_drag_and_drop**(mir::scene::Surface const *surf, std::vector<uint8_t> const &handle) = 0

virtual void **depth_layer_set_to**(mir::scene::Surface const *surf, *MirDepthLayer* depth_layer) = 0

virtual void **application_id_set_to**(mir::scene::Surface const *surf, std::string const &application_id) = 0

Class SpinnerSplash

- Defined in file_examples_miral-shell_spinner_splash.h

Class Documentation

class **SpinnerSplash**

Public Functions

SpinnerSplash()

~SpinnerSplash()

void **operator()**(struct wl_display *display)

void **operator()**(std::weak_ptr<mir::scene::Session> const &session)

operator std::shared_ptr<*SplashSession*>() const

Class SplashSession

- Defined in file_examples_example-server-lib_splash_session.h

Class Documentation

class **SplashSession**

Public Functions

virtual auto **session**() const -> std::shared_ptr<mir::scene::Session> = 0

SplashSession() = default

virtual **~SplashSession**() = default

SplashSession(*SplashSession* const&) = delete

SplashSession &**operator=**(*SplashSession* const&) = delete

Class SwSplash

- Defined in file_examples_example-server-lib_sw_splash.h

Class Documentation

class **SwSplash**

Public Functions

SwSplash()

~SwSplash()

void **operator()**(struct wl_display *display)

void **operator()**(std::weak_ptr<mir::scene::Session> const &session)

void **enable**(bool show_splash_opt)

operator std::shared_ptr<*SplashSession*>() const

Class TilingWindowManagerPolicy

- Defined in file_examples_example-server-lib_tiling_window_manager.h

Nested Relationships

Nested Types

- *Class TilingWindowManagerPolicy::MRUTileList*

Inheritance Relationships

Base Type

- `public miral::WindowManagementPolicy` (*Class WindowManagementPolicy*)

Class Documentation

class **TilingWindowManagerPolicy** : public miral::*WindowManagementPolicy*

Public Functions

explicit **TilingWindowManagerPolicy**(miral::*WindowManagerTools* const &tools,
std::shared_ptr<*SplashSession*> const &spinner,
miral::*InternalClientLauncher* const &launcher)

virtual auto **place_new_window**(miral::*ApplicationInfo* const &app_info, miral::*WindowSpecification* const &request_parameters) -> miral::*WindowSpecification* override

Customize initial window placement.

Parameters

- **app_info** – the application requesting a new window
- **requested_specification** – the requested specification (updated with default placement)

Returns

the customized specification

virtual void **handle_window_ready**(miral::*WindowInfo* &window_info) override

notification that the first buffer has been posted

Parameters

window_info – the window

virtual void **handle_modify_window**(miral::*WindowInfo* &window_info, miral::*WindowSpecification* const &modifications) override

request from client to modify the window specification.

Parameters

- **window_info** – the window
- **modifications** – the requested changes

Note: the request has already been validated against the type definition

---

virtual bool **handle_keyboard_event**(MirKeyboardEvent const *event) override

keyboard event handler

Parameters

event – the event

Returns

whether the policy has consumed the event

virtual bool **handle_touch_event**(MirTouchEvent const *event) override

touch event handler

Parameters

event – the event

Returns

whether the policy has consumed the event

virtual bool **handle_pointer_event**(MirPointerEvent const *event) override

pointer event handler

Parameters

event – the event

Returns

whether the policy has consumed the event

virtual void **handle_raise_window**(miral::*WindowInfo* &window_info) override

request from client to raise the window

Parameters

window_info – the window

Note: the request has already been validated against the requesting event

virtual void **advise_end**() override

after any related calls end

virtual void **advise_new_window**(miral::*WindowInfo* const &window_info) override

Notification that a window has been created.

Parameters

window_info – the window

virtual void **advise_focus_gained**(miral::*WindowInfo* const &info) override

Notification that a window has gained focus.

Parameters

window_info – the window

virtual void **advise_new_app**(miral::*ApplicationInfo* &application) override

Notification that a new application has connected.

Parameters

application – the application

virtual void **advise_delete_app**(miral::*ApplicationInfo* const &application) override

Notification that an application has disconnected.

Parameters

application – the application

virtual void **handle_request_move**(miral::*WindowInfo* &window_info, MirInputEvent const *input_event) override

request from client to initiate move

Parameters

- **window_info** – the window
- **input_event** – the requesting event

Note: the request has already been validated against the requesting event

virtual void **handle_request_resize**(miral::*WindowInfo* &window_info, MirInputEvent const *input_event, *MirResizeEdge* edge) override

request from client to initiate resize

Parameters

- **window_info** – the window
- **input_event** – the requesting event
- **edge** – the edge(s) being dragged

Note: the request has already been validated against the requesting event

virtual auto **confirm_inherited_move**(miral::*WindowInfo* const &window_info, Displacement movement) -> Rectangle override

Confirm (and optionally adjust) the motion of a child window when the parent is moved.

Parameters

- **window_info** – the window
- **movement** – the movement of the parent

Returns

the confirmed placement of the window

virtual Rectangle **confirm_placement_on_display**(const miral::*WindowInfo* &window_info, *MirWindowState* new_state, Rectangle const &new_placement) override

Confirm (and optionally adjust) the placement of a window on the display.

Called when (re)placing fullscreen, maximized, horizontally maximised and vertically maximized windows to allow adjustment for decorations.

Parameters

- **window_info** – the window
- **new_state** – the new state
- **new_placement** – the suggested placement

Returns

the confirmed placement of the window

Class TilingWindowManagerPolicy::MRUTileList

- Defined in file_examples_example-server-lib_tiling_window_manager.h

Nested Relationships

This class is a nested type of *Class TilingWindowManagerPolicy*.

Class Documentation

class **MRUTileList**

Public Types

using **Enumerator** = std::function<void(std::shared_ptr<void> const &tile)>

Public Functions

void **push**(std::shared_ptr<void> const &tile)

void **erase**(std::shared_ptr<void> const &tile)

void **enumerate**(*Enumerator* const &enumerator) const

inline auto **count**() -> size_t

Class WaylandApp

- Defined in file_examples_example-server-lib_wayland_app.h

Class Documentation

class **WaylandApp**

Public Functions

WaylandApp()

WaylandApp(wl_display *display)

virtual **~WaylandApp**() = default

void **wayland_init**(wl_display *display)

Needs to be two-step initialized to virtual methods are called.

void **roundtrip**() const

inline auto **display**() const -> wl_display*

inline auto **compositor**() const -> wl_compositor*

inline auto **shm**() const -> wl_shm*

inline auto **seat**() const -> wl_seat*

inline auto **shell**() const -> wl_shell*

Protected Functions

inline virtual void **output_ready**(*WaylandOutput* const*)

inline virtual void **output_changed**(*WaylandOutput* const*)

inline virtual void **output_gone**(*WaylandOutput* const*)

Protected Attributes

friend WaylandOutput

Class WaylandCallback

- Defined in file_examples_example-server-lib_wayland_app.h

Class Documentation

class **WaylandCallback**

Public Static Functions

static void **create**(wl_callback *callback, std::function<void()> &&func)
Takes ownership of callback.

Template Class WaylandObject

- Defined in file_examples_example-server-lib_wayland_app.h

Class Documentation

template<typename **T**>

class **WaylandObject**

Public Functions

inline **WaylandObject**()

inline **WaylandObject**(*T* *proxy, void (*destroy)(*T**))

inline **operator** *T**() const

Class WaylandOutput

- Defined in file_examples_example-server-lib_wayland_app.h

Class Documentation

class **WaylandOutput**

Public Functions

WaylandOutput(*WaylandApp* *app, wl_output *output)

virtual **~WaylandOutput**() = default

inline auto **scale**() const -> int

inline auto **transform**() const -> int

inline auto **operator==**(wl_output *other) const -> bool

inline auto **wl**() const -> wl_output*

Class WaylandShm

- Defined in file_examples_example-server-lib_wayland_shm.h

Class Documentation

class **WaylandShm**

A single *WaylandShm* does not efficiently provision multiple buffers for multiple window sizes. Please use one *WaylandShm* per window (if windows may have distinct sizes)

Public Functions

WaylandShm(wl_shm *shm)

Does not take ownership of the wl_shm.

auto **get_buffer**(mir::geometry::*Size* size, mir::geometry::*Stride* stride) ->
std::shared_ptr<*WaylandShmBuffer*>

Always returns a buffer of the correct size that is not in-use.

Class WaylandShmBuffer

- Defined in file_examples_example-server-lib_wayland_shm.h

Inheritance Relationships

Base Type

- `public std::enable_shared_from_this< WaylandShmBuffer >`

Class Documentation

class **WaylandShmBuffer** : public std::enable_shared_from_this<*WaylandShmBuffer*>

Public Functions

WaylandShmBuffer(std::shared_ptr<WaylandShmPool> pool, void *data, mir::geometry::*Size* size, mir::geometry::*Stride* stride, wl_buffer *buffer)

~WaylandShmBuffer()

inline auto **data**() const -> void*

inline auto **size**() const -> mir::geometry::*Size*

inline auto **stride**() const -> mir::geometry::*Stride*

inline auto **is_in_use**() const -> bool

Returns if this buffer is currently being used by the compositor. In-use buffers keep themselves alive.

auto **use**() -> wl_buffer*

Marks this buffer as in-use and assumes the resulting wl_buffer is sent to the compositor. Keeps this buffer alive until the compositor releases it (if it's not sent to the compositor or the compositor never releases it this buffer is leaked). Should only be called if the buffer is not already in-use.

Class WaylandSurface

- Defined in file_examples_example-server-lib_wayland_surface.h

Inheritance Relationships

Derived Type

- `private MirEglSurface` (*Class MirEglSurface*)

Class Documentation

class **WaylandSurface**

Subclassed by *MirEglSurface*

Public Functions

WaylandSurface(*WaylandApp* const *app)

virtual **~WaylandSurface**() = default

void **attach_buffer**(wl_buffer *buffer, int scale)

void **commit**() const

void **set_fullscreen**(wl_output *output)

output can be null, user needs to commit after

void **add_frame_callback**(std::function<void()> &&func)

inline auto **app**() const -> *WaylandApp* const*

inline auto **surface**() const -> wl_surface*

inline auto **configured_size**() const -> mir::geometry::*Size*

Protected Functions

inline virtual void **configured**()

Called when the compositor configures this shell surface.

Enums

Enum @0

- Defined in file_include_core_mir_toolkit_mir_native_buffer.h

Enum Documentation

enum **[anonymous]**

Values:

enumerator **mir_buffer_package_max**

Enum MirBufferFlag

- Defined in file_include_core_mir_toolkit_mir_native_buffer.h

Enum Documentation

enum **MirBufferFlag**

Values:

enumerator **mir_buffer_flag_can_scanout**

enumerator **mir_buffer_flag_fenced**

Enum MirDepthLayer

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirDepthLayer**

Depth layer controls Z ordering of surfaces.

A surface will always appear on top of surfaces with a lower depth layer, and below those with a higher one. A depth layer can be converted to a number with *mir::mir_depth_layer_get_index()*. This is useful for creating a list indexed by depth layer, or comparing the height of two layers.

Values:

enumerator **mir_depth_layer_background**

For desktop backgrounds and alike (lowest layer)

enumerator **mir_depth_layer_below**

For panels or other controls/decorations below normal windows.

enumerator **mir_depth_layer_application**

For normal application windows.

enumerator **mir_depth_layer_always_on_top**

For always-on-top application windows.

enumerator **mir_depth_layer_above**

For panels or notifications that want to be above normal windows.

enumerator **mir_depth_layer_overlay**

For overlays such as lock screens (heighest layer)

Enum MirEdgeAttachment

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirEdgeAttachment**

Values:

enumerator **mir_edge_attachment_vertical**

enumerator **mir_edge_attachment_horizontal**

enumerator **mir_edge_attachment_any**

Enum MirEventType

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirEventType**

Values:

enumerator **mir_event_type_key**

enumerator **mir_event_type_motion**

enumerator **mir_event_type_window**

enumerator **mir_event_type_resize**

enumerator **mir_event_type_prompt_session_state_change**

enumerator **mir_event_type_orientation**

enumerator **mir_event_type_close_window**

enumerator **mir_event_type_input**

enumerator **mir_event_type_input_configuration**

enumerator **mir_event_type_window_output**

enumerator **mir_event_type_input_device_state**

enumerator **mir_event_type_window_placement**

Enum MirFocusMode

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirFocusMode**

Focus mode controls how a surface gains and loses focus.

Values:

enumerator **mir_focus_mode_focusable**

The surface can gain and lose focus normally.

enumerator **mir_focus_mode_disabled**

The surface will never be given focus.

enumerator **mir_focus_mode_grabbing**

This mode causes the surface to take focus if possible, and prevents focus from leaving it as long as it has this mode.

Enum MirFormFactor

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirFormFactor**

Form factor associated with a physical output.

Values:

enumerator **mir_form_factor_unknown**

enumerator **mir_form_factor_phone**

enumerator **mir_form_factor_tablet**

enumerator **mir_form_factor_monitor**

enumerator **mir_form_factor_tv**

enumerator **mir_form_factor_projector**

Enum MirInputDeviceCapability

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Enum Documentation

enum **MirInputDeviceCapability**

Values:

enumerator **mir_input_device_capability_none**

enumerator **mir_input_device_capability_pointer**

enumerator **mir_input_device_capability_keyboard**

enumerator **mir_input_device_capability_touchpad**

enumerator **mir_input_device_capability_touchscreen**

enumerator **mir_input_device_capability_gamepad**

enumerator **mir_input_device_capability_joystick**

enumerator **mir_input_device_capability_switch**

enumerator **mir_input_device_capability_multitouch**

enumerator **mir_input_device_capability_alpha_numeric**

capable to detect multiple contacts

Enum MirInputEventModifier

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirInputEventModifier**

Description of key modifier state.

Values:

enumerator **mir_input_event_modifier_none**

enumerator **mir_input_event_modifier_alt**

enumerator **mir_input_event_modifier_alt_left**

enumerator **mir_input_event_modifier_alt_right**

enumerator **mir_input_event_modifier_shift**

enumerator **mir_input_event_modifier_shift_left**

enumerator **mir_input_event_modifier_shift_right**

enumerator **mir_input_event_modifier_sym**

enumerator **mir_input_event_modifier_function**

enumerator **mir_input_event_modifier_ctrl**

enumerator **mir_input_event_modifier_ctrl_left**

enumerator **mir_input_event_modifier_ctrl_right**

enumerator **mir_input_event_modifier_meta**

enumerator **mir_input_event_modifier_meta_left**

enumerator **mir_input_event_modifier_meta_right**

enumerator **mir_input_event_modifier_caps_lock**

enumerator **mir_input_event_modifier_num_lock**

enumerator **mir_input_event_modifier_scroll_lock**

Enum MirInputEventType

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirInputEventType**

Values:

enumerator **mir_input_event_type_key**

enumerator **mir_input_event_type_touch**

enumerator **mir_input_event_type_pointer**

enumerator **mir_input_event_type_keyboard_resync**

enumerator **mir_input_event_types**

Enum MirKeyboardAction

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirKeyboardAction**

Possible actions for changing key state.

Values:

enumerator **mir_keyboard_action_up**

enumerator **mir_keyboard_action_down**

enumerator **mir_keyboard_action_repeat**

enumerator **mir_keyboard_action_modifiers**

enumerator **mir_keyboard_actions**

Enum MirLifecycleState

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirLifecycleState**

Values:

enumerator **mir_lifecycle_state_will_suspend**

enumerator **mir_lifecycle_state_resumed**

enumerator **mir_lifecycle_connection_lost**

Enum MirMirrorMode

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirMirrorMode**

Mirroring axis relative to the "natural" orientation of the display.

Values:

enumerator **mir_mirror_mode_none**

enumerator **mir_mirror_mode_vertical**

enumerator **mir_mirror_mode_horizontal**

Enum MirOrientation

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirOrientation**

Direction relative to the "natural" orientation of the display.

Values:

enumerator **mir_orientation_normal**

enumerator **mir_orientation_left**

enumerator **mir_orientation_inverted**

enumerator **mir_orientation_right**

Enum MirOrientationMode

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirOrientationMode**

Values:

enumerator **mir_orientation_mode_portrait**

enumerator **mir_orientation_mode_landscape**

enumerator **mir_orientation_mode_portrait_inverted**

enumerator **mir_orientation_mode_landscape_inverted**

enumerator **mir_orientation_mode_portrait_any**

enumerator **mir_orientation_mode_landscape_any**

enumerator **mir_orientation_mode_any**

Enum MirOutputGammaSupported

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirOutputGammaSupported**

Supports gamma correction.

Values:

enumerator **mir_output_gamma_unsupported**

enumerator **mir_output_gamma_supported**

Enum MirOutputType

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirOutputType**

Values:

enumerator **mir_output_type_unknown**

enumerator **mir_output_type_vga**

enumerator **mir_output_type_dvii**

enumerator **mir_output_type_dvid**

enumerator **mir_output_type_dvia**

enumerator **mir_output_type_composite**

enumerator **mir_output_type_svideo**

enumerator **mir_output_type_lvds**

enumerator **mir_output_type_component**

enumerator **mir_output_type_ninepindin**

enumerator **mir_output_type_displayport**

enumerator **mir_output_type_hdmia**

enumerator **mir_output_type_hdmib**

enumerator **mir_output_type_tv**

enumerator **mir_output_type_edp**

enumerator **mir_output_type_virtual**

enumerator **mir_output_type_dsi**

enumerator **mir_output_type_dpi**

Enum MirPixelFormat

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirPixelFormat**

32-bit pixel formats (8888): The order of components in the enum matches the order of the components as they would be written in an integer representing a pixel value of that format.

For example; abgr_8888 should be coded as 0xAABBGGRR, which will end up as R,G,B,A in memory on a little endian system, and as A,B,G,R on a big endian system.

24-bit pixel formats (888): These are in literal byte order, regardless of CPU architecture it's always the same. Writing these 3-byte pixels is typically slower than other formats but uses less memory than 32-bit and is endian-independent.

16-bit pixel formats (565/5551/4444): Always interpreted as one 16-bit integer per pixel with components in high-to-low bit order following the format name. These are the fastest formats, however colour quality is visibly lower.

Values:

enumerator **mir_pixel_format_invalid**

enumerator **mir_pixel_format_abgr_8888**

enumerator **mir_pixel_format_xbgr_8888**

enumerator **mir_pixel_format_argb_8888**

enumerator **mir_pixel_format_xrgb_8888**

enumerator **mir_pixel_format_bgr_888**

enumerator **mir_pixel_format_rgb_888**

enumerator **mir_pixel_format_rgb_565**

enumerator **mir_pixel_format_rgba_5551**

enumerator **mir_pixel_format_rgba_4444**

enumerator **mir_pixel_formats**

Enum MirPlacementGravity

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirPlacementGravity**

Reference point for aligning a surface relative to a rectangle.

Each element (surface and rectangle) has a MirPlacementGravity assigned.

Values:

enumerator **mir_placement_gravity_center**

the reference point is at the center.

enumerator **mir_placement_gravity_west**

the reference point is at the middle of the left edge.

enumerator **mir_placement_gravity_east**

the reference point is at the middle of the right edge.

enumerator **mir_placement_gravity_north**

the reference point is in the middle of the top edge.

enumerator **mir_placement_gravity_south**

the reference point is at the middle of the lower edge.

enumerator **mir_placement_gravity_northwest**

the reference point is at the top left corner.

enumerator **mir_placement_gravity_northeast**

the reference point is at the top right corner.

enumerator **mir_placement_gravity_southwest**

the reference point is at the lower left corner.

enumerator **mir_placement_gravity_southeast**

the reference point is at the lower right corner.

Enum MirPlacementHints

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirPlacementHints**

Positioning hints for aligning a window relative to a rectangle.

These hints determine how the window should be positioned in the case that the surface would fall off-screen if placed in its ideal position.

For example, `mir_placement_hints_flip_x` will invert the x component of `aux_rect_placement_offset` and replace `mir_placement_gravity_northwest` with `mir_placement_gravity_northeast` and vice versa if the window extends beyond the left or right edges of the monitor.

If `mir_placement_hints_slide_x` is set, the window can be shifted horizontally to fit on-screen.

If `mir_placement_hints_resize_x` is set, the window can be shrunken horizontally to fit.

If `mir_placement_hints_antipodes` is set then the rect gravity may be substituted with the opposite corner (e.g. `mir_placement_gravity_northeast` to `mir_placement_gravity_southwest`) in combination with other options.

When multiple flags are set, flipping should take precedence over sliding, which should take precedence over resizing.

Values:

enumerator **mir_placement_hints_flip_x**

allow flipping anchors horizontally

enumerator **mir_placement_hints_flip_y**

allow flipping anchors vertically

enumerator **mir_placement_hints_slide_x**

allow sliding window horizontally

enumerator **mir_placement_hints_slide_y**
allow sliding window vertically

enumerator **mir_placement_hints_resize_x**
allow resizing window horizontally

enumerator **mir_placement_hints_resize_y**
allow resizing window vertically

enumerator **mir_placement_hints_antipodes**
allow flipping aux_anchor to opposite corner

enumerator **mir_placement_hints_flip_any**
allow flipping anchors on both axes

enumerator **mir_placement_hints_slide_any**
allow sliding window on both axes

enumerator **mir_placement_hints_resize_any**
allow resizing window on both axes

Enum MirPointerAcceleration

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Enum Documentation

enum **MirPointerAcceleration**
MirPointerAcceleration describes the way pointer movement is filtered:

- mir_pointer_acceleration_none: (acceleration bias + 1.0) is applied as a factor to the current velocity of the pointer. So a bias of 0 to results to no change of velocity.
- mir_pointer_acceleration_adaptive: acceleration bias selects an acceleration function based on the current velocity that usually consists of two linear inclines separated by a plateau.

Values:

enumerator **mir_pointer_acceleration_none**

enumerator **mir_pointer_acceleration_adaptive**

Enum MirPointerAction

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirPointerAction**

Possible pointer actions.

Values:

enumerator **mir_pointer_action_button_up**

enumerator **mir_pointer_action_button_down**

enumerator **mir_pointer_action_enter**

enumerator **mir_pointer_action_leave**

enumerator **mir_pointer_action_motion**

enumerator **mir_pointer_actions**

Enum MirPointerAxis

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirPointerAxis**

Identifiers for pointer axis.

Values:

enumerator **mir_pointer_axis_x**

enumerator **mir_pointer_axis_y**

enumerator **mir_pointer_axis_vscroll**

enumerator **mir_pointer_axis_hscroll**

enumerator **mir_pointer_axis_relative_x**

enumerator **mir_pointer_axis_relative_y**

enumerator **mir_pointer_axis_vscroll_discrete**

enumerator **mir_pointer_axis_hscroll_discrete**

enumerator **mir_pointer_axis_vscroll_value120**

enumerator **mir_pointer_axis_hscroll_value120**

enumerator **mir_pointer_axes**

Enum MirPointerAxisSource

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirPointerAxisSource**

Identifiers for pointer event source.

Values:

enumerator **mir_pointer_axis_source_none**

enumerator **mir_pointer_axis_source_wheel**

enumerator **mir_pointer_axis_source_finger**

enumerator **mir_pointer_axis_source_continuous**

enumerator **mir_pointer_axis_source_wheel_tilt**

Enum MirPointerButton

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirPointerButton**

Values:

enumerator **mir_pointer_button_primary**

enumerator **mir_pointer_button_secondary**

enumerator **mir_pointer_button_tertiary**

enumerator **mir_pointer_button_back**

enumerator **mir_pointer_button_forward**

enumerator **mir_pointer_button_side**

enumerator **mir_pointer_button_extra**

enumerator **mir_pointer_button_task**

Enum MirPointerConfinementState

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirPointerConfinementState**

Pointer Confinement.

Values:

enumerator **mir_pointer_unconfined**

enumerator **mir_pointer_confined_oneshot**

enumerator **mir_pointer_confined_persistent**

enumerator **mir_pointer_locked_oneshot**

enumerator **mir_pointer_locked_persistent**

Enum MirPointerHandedness

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Enum Documentation

enum **MirPointerHandedness**

Values:

enumerator **mir_pointer_handedness_right**

enumerator **mir_pointer_handedness_left**

Enum MirPowerMode

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirPowerMode**

Values:

enumerator **mir_power_mode_on**

enumerator **mir_power_mode_standby**

enumerator **mir_power_mode_suspend**

enumerator **mir_power_mode_off**

Enum MirPromptSessionState

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirPromptSessionState**

Values:

enumerator **mir_prompt_session_state_stopped**

enumerator **mir_prompt_session_state_started**

enumerator **mir_prompt_session_state_suspended**

Enum MirResizeEdge

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirResizeEdge**

Hints for resizing a window.

These values are used to indicate which edge(s) of a surface is being dragged in a resize operation.

Values:

enumerator **mir_resize_edge_none**

enumerator **mir_resize_edge_west**

enumerator **mir_resize_edge_east**

enumerator **mir_resize_edge_north**

enumerator **mir_resize_edge_south**

enumerator **mir_resize_edge_northwest**

enumerator **mir_resize_edge_northeast**

enumerator **mir_resize_edge_southwest**

enumerator **mir_resize_edge_southeast**

Enum MirShellChrome

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirShellChrome**

Shell chrome.

Values:

enumerator **mir_shell_chrome_normal**

enumerator **mir_shell_chrome_low**

Enum MirSubpixelArrangement

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirSubpixelArrangement**

Physical arrangement of subpixels on the physical output.

This is always relative to the "natural" orientation of the display - mir_orientation_normal.

Values:

enumerator **mir_subpixel_arrangement_unknown**

Arrangement of subpixels cannot be determined.

enumerator **mir_subpixel_arrangement_horizontal_rgb**

Subpixels are arranged horizontally, R, G, B from left to right.

enumerator **mir_subpixel_arrangement_horizontal_bgr**

Subpixels are arranged horizontally, B, G, R from left to right.

enumerator **mir_subpixel_arrangement_vertical_rgb**

Subpixels are arranged vertically, R, G, B from top to bottom.

enumerator **mir_subpixel_arrangement_vertical_bgr**

Subpixels are arranged vertically, B, G, R from top to bottom.

enumerator **mir_subpixel_arrangement_none**

Device does not have regular subpixels.

Enum MirTouchAction

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirTouchAction**

Possible per touch actions for state changing.

Values:

enumerator **mir_touch_action_up**

enumerator **mir_touch_action_down**

enumerator **mir_touch_action_change**

enumerator **mir_touch_actions**

Enum MirTouchAxis

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirTouchAxis**

Identifiers for touch axis.

Values:

enumerator **mir_touch_axis_x**

enumerator **mir_touch_axis_y**

enumerator **mir_touch_axis_pressure**

enumerator **mir_touch_axis_touch_major**

enumerator **mir_touch_axis_touch_minor**

enumerator **mir_touch_axis_size**

enumerator **mir_touch_axes**

Enum MirTouchpadClickMode

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Enum Documentation

enum **MirTouchpadClickMode**

MirTouchpadClickMode configures how the touchpad itself should generate pointer button events.

The available click modes may be active simultaneously.
- mir_touchpad_click_mode_none: no active click mode
- mir_touchpad_click_mode_area_to_click: simulate pointer buttons using click areas on the touchpad
- mir_touchpad_click_mode_finger_count: simulate pointer buttons using the number of fingers down

Values:

enumerator **mir_touchpad_click_mode_none**

enumerator **mir_touchpad_click_mode_area_to_click**

enumerator **mir_touchpad_click_mode_finger_count**

Enum MirTouchpadScrollMode

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Enum Documentation

enum **MirTouchpadScrollMode**

MirTouchpadScrollMode configures how the touchpad should generate scroll events.

- mir_touchpad_scroll_mode_none: no scroll
- mir_touchpad_scroll_mode_two_finger_scroll: two finger movement generates generates vertical and horizontal scroll events
- mir_touchpad_scroll_mode_edge_scroll: touch movement at the edge of the touchpad genertes scroll events
- mir_touchpad_scroll_mode_button_down_scroll: movement on the touchpad generates scroll events when a button is held down simultaneously

Values:

enumerator **mir_touchpad_scroll_mode_none**

enumerator **mir_touchpad_scroll_mode_two_finger_scroll**

enumerator **mir_touchpad_scroll_mode_edge_scroll**

enumerator **mir_touchpad_scroll_mode_button_down_scroll**

Enum MirTouchscreenMappingMode

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Enum Documentation

enum **MirTouchscreenMappingMode**

Mapping modes for touchscreen devices.

The mode defines how coordinates from the touchscreen frequently referred to as device coordinates are translated into scene coordinates.

This configuration mode is relevant for different classes of input devices, i.e handheld devices with builtin touchscreens or external graphic tablets or external monitors with touchscreen capabilities.

Values:

enumerator **mir_touchscreen_mapping_mode_to_output**

Map the device coordinates onto specific output.

enumerator **mir_touchscreen_mapping_mode_to_display_wall**

Map the device coordinates onto the whole wall of outputs.

Enum MirTouchTooltype

- Defined in file_include_core_mir_toolkit_events_enums.h

Enum Documentation

enum **MirTouchTooltype**

Identifiers for per-touch tool types.

Values:

enumerator **mir_touch_tooltype_unknown**

enumerator **mir_touch_tooltype_finger**

enumerator **mir_touch_tooltype_stylus**

enumerator **mir_touch_tooltypes**

Enum MirWindowAttrib

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirWindowAttrib**

Attributes of a window that the client and server/shell may wish to get or set over the wire.

Values:

enumerator **mir_window_attrib_type**

enumerator **mir_window_attrib_state**

enumerator **mir_window_attrib_swapinterval**

Deprecated:

Do not listen for events reporting this attribute.

Use the "mir_*_get_swapinterval()" functions instead if you wish query its value

enumerator **mir_window_attrib_focus**

enumerator **mir_window_attrib_dpi**

enumerator **mir_window_attrib_visibility**

enumerator **mir_window_attrib_preferred_orientation**

enumerator **mir_window_attribs**

Enum MirWindowFocusState

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirWindowFocusState**

Values:

enumerator **mir_window_focus_state_unfocused**

Inactive and does not have focus

enumerator **mir_window_focus_state_focused**

Active and has keybaord focus

enumerator **mir_window_focus_state_active**

Active but does not have keyboard focus

Enum MirWindowState

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirWindowState**

Values:

enumerator **mir_window_state_unknown**

enumerator **mir_window_state_restored**

enumerator **mir_window_state_minimized**

enumerator **mir_window_state_maximized**

enumerator **mir_window_state_vertmaximized**

enumerator **mir_window_state_fullscreen**

enumerator **mir_window_state_horizmaximized**

enumerator **mir_window_state_hidden**

enumerator **mir_window_state_attached**

Used for panels, notifications and other windows attached to output edges.

enumerator **mir_window_states**

Enum MirWindowType

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirWindowType**

Values:

enumerator **mir_window_type_normal**

AKA “regular”

enumerator **mir_window_type_utility**

AKA “floating”

enumerator **mir_window_type_dialog**

enumerator **mir_window_type_gloss**

enumerator **mir_window_type_freestyle**

enumerator **mir_window_type_menu**

enumerator **mir_window_type_inputmethod**

AKA “OSK” or handwriting etc.

enumerator **mir_window_type_satellite**

AKA “toolbox”/”toolbar”

enumerator **mir_window_type_tip**

AKA “tooltip”

enumerator **mir_window_type_decoration**

enumerator **mir_window_types**

Enum MirWindowVisibility

- Defined in file_include_core_mir_toolkit_common.h

Enum Documentation

enum **MirWindowVisibility**

Values:

enumerator **mir_window_visibility_occluded**

enumerator **mir_window_visibility_exposed**

Unions

Union Descriptor::Value

- Defined in file_include_miroil_miroil_edid.h

Nested Relationships

This union is a nested type of *Struct Edid::Descriptor*.

Union Documentation

union **Value**

#include <edid.h>

Public Members

char **monitor_name**[13]

char **unspecified_text**[13]

char **serial_number**[13]

Functions

Function make_mir_eglapp

- Defined in file_examples_miral-shell_spinner_miregl.h

Function Documentation

std::shared_ptr<MirEglApp> **make_mir_eglapp**(struct wl_display *display)

Function mir::fatal_error_abort

- Defined in file_include_core_mir_fatal.h

Function Documentation

void `mir::`**fatal_error_abort**(char const *reason, ...)

An alternative to *fatal_error_except()* that kills the program and dump core as cleanly as possible.

Parameters

reason – **[in]** A printf-style format string.

Function mir::fatal_error_except

- Defined in file_include_core_mir_fatal.h

Function Documentation

void `mir::`**fatal_error_except**(char const *reason, ...)

Throws an exception that will typically kill the Mir server and propagate from mir::run_mir.

Parameters

reason – **[in]** A printf-style format string.

Template Function mir::geometry::as_delta(generic::X<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*DeltaX*<*T*> `mir::geometry::`**as_delta**(generic::*X*<*T*> const &x)

Template Function mir::geometry::as_delta(generic::Y<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*DeltaY*<*T*> `mir::geometry::`**`as_delta`**(generic::*Y*<*T*> const &y)

Template Function mir::geometry::as_delta(generic::Width<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*DeltaX*<*T*> `mir::geometry::`**`as_delta`**(generic::*Width*<*T*> const &w)

Template Function mir::geometry::as_delta(generic::Height<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*DeltaY*<*T*> `mir::geometry::`**`as_delta`**(generic::*Height*<*T*> const &h)

Template Function mir::geometry::as_height(generic::DeltaY<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*Height*<*T*> `mir::geometry::`**`as_height`**(generic::*DeltaY*<*T*> const &dy)

Template Function mir::geometry::as_height(generic::Y<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*Height*<*T*> `mir::geometry::`**`as_height`**(generic::*Y*<*T*> const &y)

Template Function mir::geometry::as_width(generic::DeltaX<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*Width*<*T*> `mir::geometry::`**`as_width`**(generic::*DeltaX*<*T*> const &dx)

Template Function mir::geometry::as_width(generic::X<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*Width*<*T*> `mir::geometry::`**`as_width`**(generic::*X*<*T*> const &x)

Template Function mir::geometry::as_x(generic::DeltaX<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*X*<*T*> `mir::geometry::`**`as_x`**(generic::*DeltaX*<*T*> const &dx)

Template Function mir::geometry::as_x(generic::Width<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*X*<*T*> `mir::geometry::as_x`(generic::*Width*<*T*> const &w)

Template Function mir::geometry::as_y(generic::DeltaY<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*Y*<*T*> `mir::geometry::as_y`(generic::*DeltaY*<*T*> const &dy)

Template Function mir::geometry::as_y(generic::Height<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr generic::*Y*<*T*> `mir::geometry::as_y`(generic::*Height*<*T*> const &h)

Template Function mir::geometry::generic::as_displacement(Size<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Displacement*<*T*> `mir::geometry::generic::as_displacement`(*Size*<*T*> const &size)

Template Function mir::geometry::generic::as_displacement(Point<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Displacement*<*T*> `mir::geometry::generic::`**`as_displacement`**(*Point*<*T*> const &point)

Template Function mir::geometry::generic::as_point(Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> `mir::geometry::generic::`**`as_point`**(*Displacement*<*T*> const &disp)

Template Function mir::geometry::generic::as_point(Size<T> const&)

- Defined in file_include_core_mir_geometry_size.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> `mir::geometry::generic::`**`as_point`**(*Size*<*T*> const &size)

Template Function mir::geometry::generic::as_size(Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Size*<*T*> `mir::geometry::generic::`**`as_size`**(*Displacement*<*T*> const &disp)

Template Function mir::geometry::generic::as_size(Point<T> const&)

- Defined in file_include_core_mir_geometry_size.h

Function Documentation

template<typename **T**>
inline constexpr *Size*<*T*> `mir::geometry::generic::`**`as_size`**(*Point*<*T*> const &point)

Template Function mir::geometry::generic::intersection_of

- Defined in file_include_core_mir_geometry_rectangle.h

Function Documentation

template<typename **T**>
Rectangle<*T*> `mir::geometry::generic::`**`intersection_of`**(*Rectangle*<*T*> const &a, *Rectangle*<*T*> const &b)

Template Function mir::geometry::generic::operator!=(Displacement<T> const&, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr bool `mir::geometry::generic::`**`operator!=`**(*Displacement*<*T*> const &lhs, *Displacement*<*T*> const &rhs)

Template Function mir::geometry::generic::operator!=(Point<T> const&, Point<T> const&)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline constexpr bool `mir::geometry::generic::`**`operator!=`**(*Point*<*T*> const &lhs, *Point*<*T*> const &rhs)

Template Function mir::geometry::generic::operator!=(Rectangle<T> const&, Rectangle<T> const&)

- Defined in file_include_core_mir_geometry_rectangle.h

Function Documentation

template<typename **T**>
inline constexpr bool `mir::geometry::generic::`**`operator!=`**(*Rectangle*<*T*> const &lhs, *Rectangle*<*T*> const &rhs)

Template Function mir::geometry::generic::operator!=(Size<T> const&, Size<T> const&)

- Defined in file_include_core_mir_geometry_size.h

Function Documentation

template<typename **T**>
inline constexpr bool `mir::geometry::generic::`**`operator!=`**(*Size*<*T*> const &lhs, *Size*<*T*> const &rhs)

Template Function mir::geometry::generic::operator*(Scalar, Width<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Width*<*T*> `mir::geometry::generic::`**`operator*`**(*Scalar* scale, *Width*<*T*> const &w)

Template Function mir::geometry::generic::operator*(Scalar, Height<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Height*<*T*> `mir::geometry::generic::`**`operator*`**(*Scalar* scale, *Height*<*T*> const &h)

Template Function mir::geometry::generic::operator*(Scalar, DeltaX<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *DeltaX*<*T*> mir::geometry::generic::**operator***(*Scalar* scale, *DeltaX*<*T*> const &dx)

Template Function mir::geometry::generic::operator*(Scalar, DeltaY<T> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *DeltaY*<*T*> mir::geometry::generic::**operator***(*Scalar* scale, *DeltaY*<*T*> const &dy)

Template Function mir::geometry::generic::operator*(Width<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Width*<*T*> mir::geometry::generic::**operator***(*Width*<*T*> const &w, *Scalar* scale)

Template Function mir::geometry::generic::operator*(Height<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Height*<*T*> mir::geometry::generic::**operator***(*Height*<*T*> const &h, *Scalar* scale)

Template Function mir::geometry::generic::operator*(DeltaX<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *DeltaX*<*T*> `mir::geometry::generic::`**`operator*`**(*DeltaX*<*T*> const &dx, *Scalar* scale)

Template Function mir::geometry::generic::operator*(DeltaY<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *DeltaY*<*T*> `mir::geometry::generic::`**`operator*`**(*DeltaY*<*T*> const &dy, *Scalar* scale)

Template Function mir::geometry::generic::operator*(Scalar, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Displacement*<*T*> `mir::geometry::generic::`**`operator*`**(*Scalar* scale, *Displacement*<*T*> const &disp)

Template Function mir::geometry::generic::operator*(Displacement<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Displacement*<*T*> `mir::geometry::generic::`**`operator*`**(*Displacement*<*T*> const &disp, *Scalar* scale)

Template Function mir::geometry::generic::operator*(Scalar, Size<T> const&)

- Defined in file_include_core_mir_geometry_size.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Size*<*T*> `mir::geometry::generic::`**`operator*`**(*Scalar* scale, *Size*<*T*> const &size)

Template Function mir::geometry::generic::operator*(Size<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_size.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Size*<*T*> `mir::geometry::generic::`**`operator*`**(*Size*<*T*> const &size, *Scalar* scale)

Template Function mir::geometry::generic::operator+(DeltaX<T>, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaX*<*T*> `mir::geometry::generic::`**`operator+`**(*DeltaX*<*T*> lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator+(DeltaY<T>, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaY*<*T*> `mir::geometry::generic::`**`operator+`**(*DeltaY*<*T*> lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator+(X<T>, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *X*<*T*> `mir::geometry::generic::`**operator+**(*X*<*T*> lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator+(Y<T>, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *Y*<*T*> `mir::geometry::generic::`**operator+**(*Y*<*T*> lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator+(Width<T>, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *Width*<*T*> `mir::geometry::generic::`**operator+**(*Width*<*T*> lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator+(Height<T>, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *Height*<*T*> `mir::geometry::generic::`**operator+**(*Height*<*T*> lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator+(Width<T>, Width<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *Width*<*T*> `mir::geometry::generic::`**operator+**(*Width*<*T*> lhs, *Width*<*T*> rhs)

Template Function mir::geometry::generic::operator+(Height<T>, Height<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *Height*<*T*> `mir::geometry::generic::`**operator+**(*Height*<*T*> lhs, *Height*<*T*> rhs)

Template Function mir::geometry::generic::operator+(Displacement<T> const&, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Displacement*<*T*> `mir::geometry::generic::`**operator+**(*Displacement*<*T*> const &lhs,
Displacement<*T*> const &rhs)

Template Function mir::geometry::generic::operator+(Point<T> const&, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> `mir::geometry::generic::`**operator+**(*Point*<*T*> const &lhs, *Displacement*<*T*>
const &rhs)

Template Function mir::geometry::generic::operator+(Displacement<T> const&, Point<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> `mir::geometry::generic::`**operator+**(*Displacement*<*T*> const &lhs, *Point*<*T*> const &rhs)

Template Function mir::geometry::generic::operator+(Point<T>, DeltaX<T>)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> `mir::geometry::generic::`**operator+**(*Point*<*T*> lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator+(Point<T>, DeltaY<T>)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> `mir::geometry::generic::`**operator+**(*Point*<*T*> lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(DeltaX<T>&, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *DeltaX*<*T*> &`mir::geometry::generic::`**operator+=**(*DeltaX*<*T*> &lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(DeltaY<T>&, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *DeltaY*<*T*> &mir::geometry::generic::**operator+=**(*DeltaY*<*T*> &lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(X<T>&, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *X*<*T*> &mir::geometry::generic::**operator+=**(*X*<*T*> &lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(Y<T>&, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *Y*<*T*> &mir::geometry::generic::**operator+=**(*Y*<*T*> &lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(Width<T>&, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *Width*<*T*> &mir::geometry::generic::**operator+=**(*Width*<*T*> &lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(Height<T>&, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *Height*<*T*> &**mir::geometry::generic::operator+=**(*Height*<*T*> &lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(Width<T>&, Width<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *Width*<*T*> &**mir::geometry::generic::operator+=**(*Width*<*T*> &lhs, *Width*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(Height<T>&, Height<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *Height*<*T*> &**mir::geometry::generic::operator+=**(*Height*<*T*> &lhs, *Height*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(Point<T>&, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> &**mir::geometry::generic::operator+=**(*Point*<*T*> &lhs, *Displacement*<*T*> const &rhs)

Template Function mir::geometry::generic::operator+=(Point<T>&, DeltaX<T>)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline *Point*<*T*> &**mir::geometry::generic::operator+=**(*Point*<*T*> &lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator+=(Point<T>&, DeltaY<T>)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline *Point*<*T*> &**mir::geometry::generic::operator+=**(*Point*<*T*> &lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator-(DeltaX<T>, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaX*<*T*> **mir::geometry::generic::operator-**(*DeltaX*<*T*> lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator-(DeltaY<T>, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaY*<*T*> **mir::geometry::generic::operator-**(*DeltaY*<*T*> lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator-(DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaX*<*T*> `mir::geometry::generic::`**`operator-`**(*DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator-(DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaY*<*T*> `mir::geometry::generic::`**`operator-`**(*DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator-(X<T>, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *X*<*T*> `mir::geometry::generic::`**`operator-`**(*X*<*T*> lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator-(Y<T>, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *Y*<*T*> `mir::geometry::generic::`**`operator-`**(*Y*<*T*> lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator-(Width<T>, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *Width*<*T*> `mir::geometry::generic::`**operator-**(*Width*<*T*> lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator-(Height<T>, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *Height*<*T*> `mir::geometry::generic::`**operator-**(*Height*<*T*> lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator-(X<T>, X<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaX*<*T*> `mir::geometry::generic::`**operator-**(*X*<*T*> lhs, *X*<*T*> rhs)

Template Function mir::geometry::generic::operator-(Y<T>, Y<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaY*<*T*> `mir::geometry::generic::`**operator-**(*Y*<*T*> lhs, *Y*<*T*> rhs)

Template Function mir::geometry::generic::operator-(Width<T>, Width<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaX*<*T*> `mir::geometry::generic::`**`operator-`**(*Width*<*T*> lhs, *Width*<*T*> rhs)

Template Function mir::geometry::generic::operator-(Height<T>, Height<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline constexpr *DeltaY*<*T*> `mir::geometry::generic::`**`operator-`**(*Height*<*T*> lhs, *Height*<*T*> rhs)

Template Function mir::geometry::generic::operator-(Displacement<T> const&, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Displacement*<*T*> `mir::geometry::generic::`**`operator-`**(*Displacement*<*T*> const &lhs,
Displacement<*T*> const &rhs)

Template Function mir::geometry::generic::operator-(Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Displacement*<*T*> `mir::geometry::generic::`**`operator-`**(*Displacement*<*T*> const &rhs)

Template Function mir::geometry::generic::operator-(Point<T> const&, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> `mir::geometry::generic::`**`operator-`**(*Point*<*T*> const &lhs, *Displacement*<*T*> const &rhs)

Template Function mir::geometry::generic::operator-(Point<T> const&, Point<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Displacement*<*T*> `mir::geometry::generic::`**`operator-`**(*Point*<*T*> const &lhs, *Point*<*T*> const &rhs)

Template Function mir::geometry::generic::operator-(Point<T>, DeltaX<T>)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> `mir::geometry::generic::`**`operator-`**(*Point*<*T*> lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator-(Point<T>, DeltaY<T>)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> `mir::geometry::generic::`**`operator-`**(*Point*<*T*> lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator-=(DeltaX<T>&, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *DeltaX*<*T*> &mir::geometry::generic::**operator-=**(*DeltaX*<*T*> &lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator-=(DeltaY<T>&, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *DeltaY*<*T*> &mir::geometry::generic::**operator-=**(*DeltaY*<*T*> &lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator-=(X<T>&, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *X*<*T*> &mir::geometry::generic::**operator-=**(*X*<*T*> &lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator-=(Y<T>&, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *Y*<*T*> &mir::geometry::generic::**operator-=**(*Y*<*T*> &lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator-=(Width<T>&, DeltaX<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *Width*<*T*> &mir::geometry::generic::**operator-=**(*Width*<*T*> &lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator-=(Height<T>&, DeltaY<T>)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**>
inline *Height*<*T*> &mir::geometry::generic::**operator-=**(*Height*<*T*> &lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator-=(Point<T>&, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr *Point*<*T*> &mir::geometry::generic::**operator-=**(*Point*<*T*> &lhs, *Displacement*<*T*> const &rhs)

Template Function mir::geometry::generic::operator-=(Point<T>&, DeltaX<T>)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline *Point*<*T*> &mir::geometry::generic::**operator-=**(*Point*<*T*> &lhs, *DeltaX*<*T*> rhs)

Template Function mir::geometry::generic::operator-=(Point<T>&, DeltaY<T>)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline *Point*<*T*> &**mir::geometry::generic::operator-=**(*Point*<*T*> &lhs, *DeltaY*<*T*> rhs)

Template Function mir::geometry::generic::operator/(Width<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Width*<*T*> **mir::geometry::generic::operator/**(*Width*<*T*> const &w, *Scalar* scale)

Template Function mir::geometry::generic::operator/(Height<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Height*<*T*> **mir::geometry::generic::operator/**(*Height*<*T*> const &h, *Scalar* scale)

Template Function mir::geometry::generic::operator/(DeltaX<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *DeltaX*<*T*> **mir::geometry::generic::operator/**(*DeltaX*<*T*> const &dx, *Scalar* scale)

Template Function mir::geometry::generic::operator/(DeltaY<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *DeltaY*<*T*> `mir::geometry::generic::`**operator/**(*DeltaY*<*T*> const &dy, *Scalar* scale)

Template Function mir::geometry::generic::operator/(Size<T> const&, Scalar)

- Defined in file_include_core_mir_geometry_size.h

Function Documentation

template<typename **T**, typename **Scalar**>
inline constexpr *Size*<*T*> `mir::geometry::generic::`**operator/**(*Size*<*T*> const &size, *Scalar* scale)

Template Function mir::geometry::generic::operator<

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline bool `mir::geometry::generic::`**operator<**(*Displacement*<*T*> const &lhs, *Displacement*<*T*> const &rhs)

Template Function mir::geometry::generic::operator<<(std::ostream&, Value<T, Tag> const&)

- Defined in file_include_core_mir_geometry_dimensions.h

Function Documentation

template<typename **T**, typename **Tag**>
std::ostream &`mir::geometry::generic::`**operator<<**(std::ostream &out, *Value*<*T*, *Tag*> const &value)

Template Function mir::geometry::generic::operator<<(std::ostream&, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
std::ostream &`mir::geometry::generic::`**`operator<<`**(std::ostream &out, *Displacement*<*T*> const &value)

Template Function mir::geometry::generic::operator<<(std::ostream&, Point<T> const&)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
std::ostream &`mir::geometry::generic::`**`operator<<`**(std::ostream &out, *Point*<*T*> const &value)

Template Function mir::geometry::generic::operator<<(std::ostream&, Rectangle<T> const&)

- Defined in file_include_core_mir_geometry_rectangle.h

Function Documentation

template<typename **T**>
std::ostream &`mir::geometry::generic::`**`operator<<`**(std::ostream &out, *Rectangle*<*T*> const &value)

Template Function mir::geometry::generic::operator<<(std::ostream&, Size<T> const&)

- Defined in file_include_core_mir_geometry_size.h

Function Documentation

template<typename **T**>
std::ostream &`mir::geometry::generic::`**`operator<<`**(std::ostream &out, *Size*<*T*> const &value)

Template Function mir::geometry::generic::operator==(Displacement<T> const&, Displacement<T> const&)

- Defined in file_include_core_mir_geometry_displacement.h

Function Documentation

template<typename **T**>
inline constexpr bool `mir::geometry::generic::`**`operator==`**(*Displacement*<*T*> const &lhs, *Displacement*<*T*> const &rhs)

Template Function mir::geometry::generic::operator==(Point<T> const&, Point<T> const&)

- Defined in file_include_core_mir_geometry_point.h

Function Documentation

template<typename **T**>
inline constexpr bool `mir::geometry::generic::`**`operator==`**(*Point*<*T*> const &lhs, *Point*<*T*> const &rhs)

Template Function mir::geometry::generic::operator==(Rectangle<T> const&, Rectangle<T> const&)

- Defined in file_include_core_mir_geometry_rectangle.h

Function Documentation

template<typename **T**>
inline constexpr bool `mir::geometry::generic::`**`operator==`**(*Rectangle*<*T*> const &lhs, *Rectangle*<*T*> const &rhs)

Template Function mir::geometry::generic::operator==(Size<T> const&, Size<T> const&)

- Defined in file_include_core_mir_geometry_size.h

Function Documentation

template<typename **T**>
inline constexpr bool `mir::geometry::generic::`**`operator==`**(*Size*<*T*> const &lhs, *Size*<*T*> const &rhs)

Function mir::geometry::operator<<

- Defined in file_include_core_mir_geometry_rectangles.h

Function Documentation

std::ostream &**mir::geometry::operator<<**(std::ostream &out, *Rectangles* const &value)

Function mir::mir_depth_layer_get_index

- Defined in file_include_core_mir_depth_layer.h

Function Documentation

auto **mir::mir_depth_layer_get_index**(*MirDepthLayer* depth_layer) -> unsigned int

Returns the height of a MirDepthLayer.

As the name implies, the returned value is usable as an array index (0 is returned for the bottommost layer and there are no gaps between layers). The values returned for each layer are in no way stable across Mir versions, and are only meaningful relative to each other.

Template Function mir::operator!=(IntWrapper<Tag, ValueType> const&, IntWrapper<Tag, ValueType> const&)

- Defined in file_include_core_mir_int_wrapper.h

Function Documentation

template<typename **Tag**, typename **ValueType**>
inline constexpr bool **mir::operator!=**(*IntWrapper*<*Tag*, *ValueType*> const &lhs, *IntWrapper*<*Tag*, *ValueType*> const &rhs)

Template Function mir::operator!=(optional_value<T> const&, optional_value<T> const&)

- Defined in file_include_core_mir_optional_value.h

Function Documentation

template<typename **T**>
inline bool **mir::operator!=**(*optional_value*<*T*> const &lhs, *optional_value*<*T*> const &rhs)

Template Function mir::operator!=(optional_value<T> const&, T const&)

- Defined in file_include_core_mir_optional_value.h

Function Documentation

template<typename **T**>
inline bool `mir::operator!=`(*optional_value*<*T*> const &lhs, *T* const &rhs)

Template Function mir::operator!=(T const&, optional_value<T> const&)

- Defined in file_include_core_mir_optional_value.h

Function Documentation

template<typename **T**>
inline bool `mir::operator!=`(*T* const &lhs, *optional_value*<*T*> const &rhs)

Template Function mir::operator<

- Defined in file_include_core_mir_int_wrapper.h

Function Documentation

template<typename **Tag**, typename **ValueType**>
inline constexpr bool `mir::operator<`(*IntWrapper*<*Tag*, *ValueType*> const &lhs, *IntWrapper*<*Tag*, *ValueType*> const &rhs)

Template Function mir::operator<<

- Defined in file_include_core_mir_int_wrapper.h

Function Documentation

template<typename **Tag**, typename **ValueType**>
std::ostream &`mir::operator<<`(std::ostream &out, *IntWrapper*<*Tag*, *ValueType*> const &value)

Template Function mir::operator<=

- Defined in file_include_core_mir_int_wrapper.h

Function Documentation

template<typename **Tag**, typename **ValueType**>
inline constexpr bool `mir::operator<=`(*IntWrapper*<*Tag*, *ValueType*> const &lhs, *IntWrapper*<*Tag*, *ValueType*> const &rhs)

Template Function mir::operator==(IntWrapper<Tag, ValueType> const&, IntWrapper<Tag, ValueType> const&)

- Defined in file_include_core_mir_int_wrapper.h

Function Documentation

template<typename **Tag**, typename **ValueType**>
inline constexpr bool `mir::operator==`(*IntWrapper*<*Tag*, *ValueType*> const &lhs, *IntWrapper*<*Tag*, *ValueType*> const &rhs)

Template Function mir::operator==(optional_value<T> const&, optional_value<T> const&)

- Defined in file_include_core_mir_optional_value.h

Function Documentation

template<typename **T**>
inline bool `mir::operator==`(*optional_value*<*T*> const &lhs, *optional_value*<*T*> const &rhs)

Template Function mir::operator==(optional_value<T> const&, T const&)

- Defined in file_include_core_mir_optional_value.h

Function Documentation

template<typename **T**>
inline bool `mir::operator==`(*optional_value*<*T*> const &lhs, *T* const &rhs)

Template Function mir::operator==(T const&, optional_value<T> const&)

- Defined in file_include_core_mir_optional_value.h

Function Documentation

template<typename **T**>
inline bool **mir::operator==**(*T* const &lhs, *optional_value*<*T*> const &rhs)

Template Function mir::operator>=

- Defined in file_include_core_mir_int_wrapper.h

Function Documentation

template<typename **Tag**, typename **ValueType**>
inline constexpr bool **mir::operator>=**(*IntWrapper*<*Tag*, *ValueType*> const &lhs, *IntWrapper*<*Tag*, *ValueType*> const &rhs)

Function mir_eglapp_init

- Defined in file_examples_miral-shell_spinner_eglapp.h

Function Documentation

std::vector<std::shared_ptr<*MirEglSurface*>> **mir_eglapp_init**(struct wl_display *display)

Function mir_surface_init

- Defined in file_examples_miral-shell_spinner_miregl.h

Function Documentation

std::vector<std::shared_ptr<*MirEglSurface*>> **mir_surface_init**(std::shared_ptr<MirEglApp> const &app)

Template Function miral::add_window_manager_policy

- Defined in file_include_miral_miral_window_management_options.h

Function Documentation

template<typename **Policy**, typename ...**Args**>
inline auto `miral::add_window_manager_policy`(std::string const &name, *Args*&... args) -> *WindowManagerOption*

Function miral::application_for(wl_client *)

- Defined in file_include_miral_miral_wayland_extensions.h

Function Documentation

auto `miral::application_for`(wl_client *client) -> *Application*

Get the MirAL application for a wl_client.

Remark

Since MirAL 2.5

Returns

The application (null if no application is found)

Function miral::application_for(wl_resource *)

- Defined in file_include_miral_miral_wayland_extensions.h

Function Documentation

auto `miral::application_for`(wl_resource *resource) -> *Application*

Get the MirAL application for a wl_resource.

Remark

Since MirAL 2.5

Returns

The application (null if no application is found)

Function miral::apply_lifecycle_state_to

- Defined in file_include_miral_miral_application.h

Function Documentation

void `miral::`**`apply_lifecycle_state_to`**(*Application* const &application, *MirLifecycleState* state)

Function miral::display_configuration_options

- Defined in file_include_miral_miral_display_configuration_option.h

Function Documentation

void `miral::`**`display_configuration_options`**(mir::Server &server)

Enable the display configuration options.

- display-config {clone,sidebyside,single,static=<filename>}
- translucent {on,off}

Function miral::equivalent_display_area

- Defined in file_include_miral_miral_output.h

Function Documentation

auto `miral::`**`equivalent_display_area`**(*Output* const &lhs, *Output* const &rhs) -> bool

Function miral::kill

- Defined in file_include_miral_miral_application.h

Function Documentation

void `miral::`**`kill`**(*Application* const &application, int sig)

Template Function miral::lambda_as_function

- Defined in file_include_miral_miral_lambda_as_function.h

Function Documentation

template<typename **Lambda**>
auto `miral::`**`lambda_as_function`**`(`*Lambda* &&lambda`)` -> typename
detail::*FunctionType*<decltype(&*Lambda*::operator**()**)>::type

Function miral::name_of

- Defined in file_include_miral_miral_application.h

Function Documentation

auto `miral::`**`name_of`**`(`*Application* const &application`)` -> std::string

Function miral::operator!=(Output::PhysicalSizeMM const&, Output::PhysicalSizeMM const&)

- Defined in file_include_miral_miral_output.h

Function Documentation

inline bool `miral::`**`operator!=`**`(`*Output*::*PhysicalSizeMM* const &lhs, *Output*::*PhysicalSizeMM* const &rhs)

Function miral::operator!=(Window const&, Window const&)

- Defined in file_include_miral_miral_window.h

Function Documentation

inline bool `miral::`**`operator!=`**`(`*Window* const &lhs, *Window* const &rhs)

Function miral::operator!=(std::shared_ptr<mir::scene::Surface> const&, Window const&)

- Defined in file_include_miral_miral_window.h

Function Documentation

inline bool `miral::operator!=`(std::shared_ptr<mir::scene::Surface> const &lhs, *Window* const &rhs)

Function miral::operator!=(Window const&, std::shared_ptr<mir::scene::Surface> const&)

- Defined in file_include_miral_miral_window.h

Function Documentation

inline bool `miral::operator!=`(*Window* const &lhs, std::shared_ptr<mir::scene::Surface> const &rhs)

Function miral::operator<

- Defined in file_include_miral_miral_window.h

Function Documentation

bool `miral::operator<`(*Window* const &lhs, *Window* const &rhs)

Function miral::operator<=

- Defined in file_include_miral_miral_window.h

Function Documentation

inline bool `miral::operator<=`(*Window* const &lhs, *Window* const &rhs)

Function miral::operator==(Output::PhysicalSizeMM const&, Output::PhysicalSizeMM const&)

- Defined in file_include_miral_miral_output.h

Function Documentation

bool `miral::operator==`(*Output*::*PhysicalSizeMM* const &lhs, *Output*::*PhysicalSizeMM* const &rhs)

Function miral::operator==(Window const&, Window const&)

- Defined in file_include_miral_miral_window.h

Function Documentation

bool `miral::operator==`(*Window* const &lhs, *Window* const &rhs)

Function miral::operator==(std::shared_ptr<mir::scene::Surface> const&, Window const&)

- Defined in file_include_miral_miral_window.h

Function Documentation

bool `miral::operator==`(std::shared_ptr<mir::scene::Surface> const &lhs, *Window* const &rhs)

Function miral::operator==(Window const&, std::shared_ptr<mir::scene::Surface> const&)

- Defined in file_include_miral_miral_window.h

Function Documentation

bool `miral::operator==`(*Window* const &lhs, std::shared_ptr<mir::scene::Surface> const &rhs)

Function miral::operator>

- Defined in file_include_miral_miral_window.h

Function Documentation

inline bool `miral::operator>`(*Window* const &lhs, *Window* const &rhs)

Function miral::operator>=

- Defined in file_include_miral_miral_window.h

Function Documentation

inline bool `miral::operator>=`(*Window* const &lhs, *Window* const &rhs)

Function miral::pid_of

- Defined in file_include_miral_miral_application.h

Function Documentation

auto `miral::pid_of`(*Application* const &application) -> pid_t

Function miral::pre_init

- Defined in file_include_miral_miral_configuration_option.h

Function Documentation

auto `miral::pre_init`(*ConfigurationOption* const &clo) -> *ConfigurationOption*

Update the option to be called back *before* Mir initialization starts.

Parameters

`clo` – the option

Function miral::PrintTo

- Defined in file_include_miral_miral_window.h

Function Documentation

void `miral::PrintTo`(*Window* const &bar, std::ostream *os)

Customization for Google test (to print surface name in errors)

See also:

https://github.com/google/googletest/blob/main/docs/advanced.md#teaching-googletest-how-to-print-your-values

---

Remark

Since MirAL 3.3

Template Function miral::set_window_management_policy

- Defined in file_include_miral_miral_set_window_management_policy.h

Function Documentation

template<typename **Policy**, typename ...**Args**>
auto `miral::`**`set_window_management_policy`**`(`*Args*`&... args) -> `*SetWindowManagementPolicy*

Function miral::socket_fd_of

- Defined in file_include_miral_miral_application.h

Function Documentation

auto `miral::`**`socket_fd_of`**`(`*Application* `const &application) -> int`

Returns the file descriptor of the client's socket connection, or -1 if there is no client socket. May be used for authentication with apparmor. X11 apps always return -1, since they do not connect directly to the Mir process.

Remark

Since MirAL 3.4

Function miral::toolkit::mir_event_get_input_event

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirInputEvent const *`miral::toolkit::`**`mir_event_get_input_event`**`(`*MirEvent* `const *event)`

Retrieve the MirInputEvent associated with a MirEvent of type mir_event_type_input.

See <mir_toolkit/events/input/input_event.h> for accessors.

Parameters

event – **[in]** The event

Returns

The associated MirInputEvent

Function miral::toolkit::mir_event_get_type

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirEventType miral::toolkit::**mir_event_get_type**(*MirEvent* const *event)

Retrieves the type of a MirEvent.

Now preferred over direct access to ev->type. In particular ev->type will never be mir_event_type_input and mir_event_get_type is the only way to ensure mir_event_get_input_event will succeed.

Parameters

event – **[in]** The event

Returns

The event type

Function miral::toolkit::mir_input_event_get_event

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirEvent const *miral::toolkit::**mir_input_event_get_event**(MirInputEvent const *event)

**

Retrieve the MirEvent associated with a given input event.

Parameters

event – **[in]** The input event

Returns

The MirEvent

Function miral::toolkit::mir_input_event_get_event_time

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

int64_t miral::toolkit::**mir_input_event_get_event_time**(MirInputEvent const *event)

**

Retrieve the time at which an input event occurred.

Parameters

event – **[in]** The input event

Returns

A timestamp in nanoseconds-since-epoch

Function miral::toolkit::mir_input_event_get_keyboard_event

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirKeyboardEvent const *miral::toolkit::**mir_input_event_get_keyboard_event**(MirInputEvent const *event)

Retrieve the MirKeyboardEvent associated with a given input event.

Parameters

event – **[in]** The input event

Returns

The MirKeyboardEvent or NULL if event type is not mir_input_event_type_key

Function miral::toolkit::mir_input_event_get_pointer_event

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirPointerEvent const *miral::toolkit::**mir_input_event_get_pointer_event**(MirInputEvent const *event)

Retrieve the MirPointerEvent associated with a given input event.

Parameters

event – **[in]** The input event

Returns

The MirPointerEvent or NULL if event type is not mir_input_event_type_pointer

Function miral::toolkit::mir_input_event_get_touch_event

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirTouchEvent const *miral::toolkit::**mir_input_event_get_touch_event**(MirInputEvent const *event)

Retrieve the MirTouchEvent associated with a given input event.

Parameters

event – **[in]** The input event

Returns

The MirTouchEvent or NULL if event type is not mir_input_event_type_touch

Function miral::toolkit::mir_input_event_get_type

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirInputEventType `miral::toolkit::`**`mir_input_event_get_type`**(MirInputEvent const *event)

Retrieve the type of an input event.

E.g. key, touch…

Parameters

`event` – **[in]** The input event

Returns

The input event type

Function miral::toolkit::mir_input_event_has_cookie

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

bool `miral::toolkit::`**`mir_input_event_has_cookie`**(MirInputEvent const *ev)

Query if an input event contains a cookie.

Parameters

`ev` – **[in]** The input event

Returns

True if the input event contains a cookie

Function miral::toolkit::mir_keyboard_event_action

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirKeyboardAction `miral::toolkit::`**`mir_keyboard_event_action`**(MirKeyboardEvent const *event)

Retrieve the action which triggered a given key event.

Parameters

`event` – **[in]** The key event

Returns

The associated action

Function miral::toolkit::mir_keyboard_event_input_event

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirInputEvent const *miral::toolkit::**mir_keyboard_event_input_event**(MirKeyboardEvent const *event)

Retrieve the corresponding input event.

Parameters

event – **[in]** The keyboard event

Returns

The input event

Function miral::toolkit::mir_keyboard_event_key_text

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

char const *miral::toolkit::**mir_keyboard_event_key_text**(MirKeyboardEvent const *event)

Retrieve the text the key press would emit as null terminated utf8 string.

The text will only be available to key down and key repeat events. For mir_keyboard_action_up or key presses that do produce text an empty string will be returned.

Parameters

event – **[in]** The key event

Returns

The text

Function miral::toolkit::mir_keyboard_event_keysym

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

xkb_keysym_t miral::toolkit::**mir_keyboard_event_keysym**(MirKeyboardEvent const *event)

Retrieve the xkb mapped keysym associated with the key acted on.

. May be interpreted as per <xkbcommon/xkbcommon-keysyms.h>

Remark

Since MirAL 3.3

Parameters

event – **[in]** The key event

Returns
The xkb_keysym

Function miral::toolkit::mir_keyboard_event_modifiers

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirInputEventModifiers `miral::toolkit::`**`mir_keyboard_event_modifiers`**(MirKeyboardEvent const *event)

Retrieve the modifier keys pressed when the key action occured.

Parameters
`event` – **[in]** The key event

Returns
The modifier mask

Function miral::toolkit::mir_keyboard_event_scan_code

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

int `miral::toolkit::`**`mir_keyboard_event_scan_code`**(MirKeyboardEvent const *event)

Retrieve the raw hardware scan code associated with the key acted on.

May be interpreted as per <linux/input.h>

Parameters
`event` – **[in]** The key event

Returns
The scancode

Function miral::toolkit::mir_pointer_event_action

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirPointerAction `miral::toolkit::`**`mir_pointer_event_action`**(MirPointerEvent const *event)

Retrieve the action which occured to generate a given pointer event.

Parameters
`event` – **[in]** The pointer event

Returns
Action performed by the pointer

Function miral::toolkit::mir_pointer_event_axis_value

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

float `miral::toolkit::`**`mir_pointer_event_axis_value`**(MirPointerEvent const *event, *MirPointerAxis* axis)

Retrieve the axis value reported by a given pointer event.

Parameters

- **`event`** – **[in]** The pointer event
- **`axis`** – **[in]** The axis to retreive a value from

Returns

The value of the given axis

Function miral::toolkit::mir_pointer_event_button_state

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

bool `miral::toolkit::`**`mir_pointer_event_button_state`**(MirPointerEvent const *event, *MirPointerButton* button)

Retrieve the state of a given pointer button when the action occurred.

Parameters

- **`event`** – **[in]** The pointer event
- **`button`** – **[in]** The button to check

Returns

Whether the given button is depressed

Function miral::toolkit::mir_pointer_event_buttons

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirPointerButtons `miral::toolkit::`**`mir_pointer_event_buttons`**(MirPointerEvent const *event)

Retreive the pointer button state as a masked set of values.

Parameters

`event` – **[in]** The pointer event

Returns

The button state

Function miral::toolkit::mir_pointer_event_input_event

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirInputEvent const *`miral::toolkit::`**`mir_pointer_event_input_event`**(MirPointerEvent const *event)

Retrieve the corresponding input event.

Parameters

`event` – **[in]** The pointer event

Returns

The input event

Function miral::toolkit::mir_pointer_event_modifiers

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirInputEventModifiers `miral::toolkit::`**`mir_pointer_event_modifiers`**(MirPointerEvent const *event)

Retrieve the modifier keys pressed when the pointer action occured.

Parameters

`event` – **[in]** The pointer event

Returns

The modifier mask

Function miral::toolkit::mir_touch_event_action

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirTouchAction `miral::toolkit::`**`mir_touch_event_action`**(MirTouchEvent const *event, unsigned int touch_index)

Retrieve the action which occured for a touch at given index.

Parameters

- **`event`** – **[in]** The touch event
- **`touch_index`** – **[in]** The touch index. Must be less than (touch_count - 1).

Returns

Action performed for the touch at index.

Function miral::toolkit::mir_touch_event_axis_value

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

float miral::toolkit::**mir_touch_event_axis_value**(MirTouchEvent const *event, unsigned int touch_index, *MirTouchAxis* axis)

Retrieve the axis value for a given axis on an indexed touch.

Parameters

- **event** – **[in]** The touch event
- **touch_index** – **[in]** The touch index. Must be less than (touch_count - 1).
- **axis** – **[in]** The axis to retreive a value from

Returns

The value of the given axis

Function miral::toolkit::mir_touch_event_id

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirTouchId miral::toolkit::**mir_touch_event_id**(MirTouchEvent const *event, unsigned int touch_index)

Retrieve the TouchID for a touch at given index.

Parameters

- **event** – **[in]** The touch event
- **touch_index** – **[in]** The touch index. Must be less than (touch_count - 1).

Returns

ID of the touch at index

Function miral::toolkit::mir_touch_event_input_event

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirInputEvent const *miral::toolkit::**mir_touch_event_input_event**(MirTouchEvent const *event)

Retrieve the corresponding input event.

Parameters

event – **[in]** The touch event

Returns

The input event

Function miral::toolkit::mir_touch_event_modifiers

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirInputEventModifiers `miral::toolkit::`**`mir_touch_event_modifiers`**(MirTouchEvent const *event)

Retrieve the modifier keys pressed when the touch action occured.

Parameters

`event` – **[in]** The key event

Returns

The modifier mask

Function miral::toolkit::mir_touch_event_point_count

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

unsigned int `miral::toolkit::`**`mir_touch_event_point_count`**(MirTouchEvent const *event)

Retrieve the number of touches reported for a given touch event.

Each touch is said to be index in the event and may be accessed by index 0, 1, … , (touch_count - 1)

Parameters

`event` – **[in]** The touch event

Returns

The number of touches

Function miral::toolkit::mir_touch_event_tooltype

- Defined in file_include_miral_miral_toolkit_event.h

Function Documentation

MirTouchTooltype `miral::toolkit::`**`mir_touch_event_tooltype`**(MirTouchEvent const *event, unsigned int touch_index)

Retrieve the tooltype for touch at given index.

Parameters

- **`event`** – **[in]** The touch event
- **`touch_index`** – **[in]** The touch index. Must be less than (touch_count - 1).

Returns

Tooltype used for the touch at index

Function miral::window_for

- Defined in file_include_miral_miral_wayland_extensions.h

Function Documentation

auto `miral::window_for`(wl_resource *surface) -> *Window*

Get the MirAL *Window* for a Wayland Surface, XdgSurface, etc. Note that there may not be a corresponding *miral::Window* (e.g. the surface is created and assigned properties before 'commit' creates the *miral::Window*).

Remark

Since MirAL 2.5

Returns

The window (null if no window is found)

Function miroil::dispatch_input_event

- Defined in file_include_miroil_miroil_eventdispatch.h

Function Documentation

void `miroil::dispatch_input_event`(const miral::*Window* &window, const MirInputEvent *event)

Specialized Template Function std::swap

- Defined in file_include_miral_miral_window_info.h

Function Documentation

template<>
inline void `std::swap`(miral::*WindowInfo* &lhs, miral::*WindowInfo* &rhs)

Function wallpaper::font_file(std::string const&)

- Defined in file_examples_example-server-lib_wallpaper_config.h

Function Documentation

void `wallpaper::font_file`(std::string const &font_file)

Function wallpaper::font_file()

- Defined in file_examples_example-server-lib_wallpaper_config.h

Function Documentation

auto `wallpaper::font_file`() -> std::string

Variables

Variable mir::fatal_error

- Defined in file_include_core_mir_fatal.h

Variable Documentation

void (*`mir::fatal_error`)(char const *reason, ...)

fatal_error() is strictly for "this should never happen" situations that you cannot recover from.

By default it points at *fatal_error_abort()*. Note the reason parameter is a simple char* so its value is clearly visible in stack trace output.

Remark

There is no attempt to make this thread-safe, if it needs to be changed that should be done before spinning up the Mir server.

Param reason
[in] A printf-style format string.

Variable mir_eglapp_background_opacity

- Defined in file_examples_miral-shell_spinner_eglapp.h

Variable Documentation

float **mir_eglapp_background_opacity**

Defines

Define __has_extension

- Defined in file_include_core_mir_toolkit_common.h

Define Documentation

__has_extension

Define __has_feature

- Defined in file_include_core_mir_toolkit_common.h

Define Documentation

__has_feature(x)

Define MIR_BYTES_PER_PIXEL

- Defined in file_include_core_mir_toolkit_common.h

Define Documentation

MIR_BYTES_PER_PIXEL(f)

Define MIR_VERSION_NUMBER

- Defined in file_include_core_mir_toolkit_mir_version_number.h

Define Documentation

MIR_VERSION_NUMBER(major, minor, micro)

MIR_VERSION_NUMBER.

Returns the combined version information as a single 32-bit value for logical comparisons. For example: #if MIR_CLIENT_VERSION >= *MIR_VERSION_NUMBER(2,3,4)*

This can be useful to conditionally build code depending on new features or specific bugfixes in the Mir client library.

Parameters

- **major** – [in] The major version (eg: 3 for version 3.2.33)
- **minor** – [in] The minor version (eg: 2 for version 3.2.33)
- **micro** – [in] The micro version (eg: 33 for version 3.2.33)

Define MIRAL_MAJOR_VERSION

- Defined in file_include_miral_miral_version.h

Define Documentation

MIRAL_MAJOR_VERSION

MIRAL_MAJOR_VERSION.

The major MirAL API version. This will increase once per API incompatible release.

See also: http://semver.org/

Define MIRAL_MICRO_VERSION

- Defined in file_include_miral_miral_version.h

Define Documentation

MIRAL_MICRO_VERSION

MIRAL_MICRO_VERSION.

The micro miral API version. This will increment each release. This is usually uninteresting for server code, but may be of use in selecting whether to use a feature that has previously been buggy.

This corresponds to the PATCH field of http://semver.org/

Define MIRAL_MINOR_VERSION

- Defined in file_include_miral_miral_version.h

Define Documentation

MIRAL_MINOR_VERSION

MIRAL_MINOR_VERSION.

The minor MirAL API version. This will increase each time new backwards-compatible API is added, and will reset to 0 each time MIRAL_MAJOR_VERSION is incremented.

See also: http://semver.org/

Define MIRAL_VERSION

- Defined in file_include_miral_miral_version.h

Define Documentation

MIRAL_VERSION

MIRAL_VERSION.

The current version of the MirAL headers in use.

Typedefs

Typedef mir::EventUPtr

- Defined in file_include_miroil_miroil_event_builder.h

Typedef Documentation

typedef std::unique_ptr<*MirEvent*, void (*)(*MirEvent**)> **mir::EventUPtr**

Typedef mir::geometry::DeltaX

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using **mir::geometry::DeltaX** = generic::*DeltaX*<int>

Typedef mir::geometry::DeltaXF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using **mir::geometry::DeltaXF** = generic::*DeltaX*<float>

Typedef mir::geometry::DeltaY

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::DeltaY` = generic::*DeltaY*<int>

Typedef mir::geometry::DeltaYF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::DeltaYF` = generic::*DeltaY*<float>

Typedef mir::geometry::Displacement

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::Displacement` = generic::*Displacement*<int>

Typedef mir::geometry::DisplacementF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::DisplacementF` = generic::*Displacement*<float>

Typedef mir::geometry::generic::DeltaX

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

template<typename **T**>

using `mir::geometry::generic::`**`DeltaX`** = *Value*<*T*, *DeltaXTag*>

Typedef mir::geometry::generic::DeltaY

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

template<typename **T**>

using `mir::geometry::generic::`**`DeltaY`** = *Value*<*T*, *DeltaYTag*>

Typedef mir::geometry::generic::Height

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

template<typename **T**>

using `mir::geometry::generic::`**`Height`** = *Value*<*T*, *HeightTag*>

Typedef mir::geometry::generic::Width

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

template<typename **T**>

using `mir::geometry::generic::`**`Width`** = *Value*<*T*, *WidthTag*>

Typedef mir::geometry::generic::X

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

template<typename **T**>

using `mir::geometry::generic::`**`X`** = *Value*<*T*, *XTag*>

Typedef mir::geometry::generic::Y

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

template<typename **T**>

using `mir::geometry::generic::`**`Y`** = *Value*<*T*, *YTag*>

Typedef mir::geometry::Height

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::`**`Height`** = generic::*Height*<int>

Typedef mir::geometry::HeightF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::`**`HeightF`** = generic::*Height*<float>

Typedef mir::geometry::Point

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::Point` = generic::*Point*<int>

Typedef mir::geometry::PointF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::PointF` = generic::*Point*<float>

Typedef mir::geometry::Rectangle

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::Rectangle` = generic::*Rectangle*<int>

Typedef mir::geometry::RectangleF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::RectangleF` = generic::*Rectangle*<float>

Typedef mir::geometry::Size

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::Size` = generic::*Size*<int>

Typedef mir::geometry::SizeF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::SizeF` = generic::*Size*<float>

Typedef mir::geometry::Stride

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::Stride` = generic::*Value*<int, *StrideTag*>

Typedef mir::geometry::Width

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::Width` = generic::*Width*<int>

Typedef mir::geometry::WidthF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::WidthF` = generic::*Width*<float>

Typedef mir::geometry::X

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::X` = generic::*X*<int>

Typedef mir::geometry::XF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::XF` = generic::*X*<float>

Typedef mir::geometry::Y

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::Y` = generic::*Y*<int>

Typedef mir::geometry::YF

- Defined in file_include_core_mir_geometry_forward.h

Typedef Documentation

using `mir::geometry::YF` = generic::*Y*<float>

Typedef miral::Application

- Defined in file_include_miral_miral_application.h

Typedef Documentation

using `miral::Application` = std::shared_ptr<mir::scene::Session>

Typedef miral::BufferStreamId

- Defined in file_include_miral_miral_window_specification.h

Typedef Documentation

typedef mir::*IntWrapper*<detail::SessionsBufferStreamIdTag> `miral::BufferStreamId`

Typedef miral::CommandLineOption

- Defined in file_include_miral_miral_command_line_option.h

Typedef Documentation

using `miral::CommandLineOption` = *ConfigurationOption*

Typedef miral::WindowManagementPolicyBuilder

- Defined in file_include_miral_miral_window_management_options.h

Typedef Documentation

using `miral::WindowManagementPolicyBuilder` = std::function<std::unique_ptr<miral::*WindowManagementPolicy*>(*WindowManagerTools* const &tools)>

Typedef MirBufferPackage

- Defined in file_include_core_mir_toolkit_mir_native_buffer.h

Typedef Documentation

typedef struct *MirBufferPackage* `MirBufferPackage`

Typedef MirClientFdCallback

- Defined in file_include_miroil_miroil_mir_prompt_session.h

Typedef Documentation

typedef void (*`MirClientFdCallback`)(*MirPromptSession* *prompt_session, size_t count, int const *fds, void *context)

Typedef MirDepthLayer

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirDepthLayer* `MirDepthLayer`

Depth layer controls Z ordering of surfaces.

A surface will always appear on top of surfaces with a lower depth layer, and below those with a higher one. A depth layer can be converted to a number with *mir::mir_depth_layer_get_index()*. This is useful for creating a list indexed by depth layer, or comparing the height of two layers.

Typedef MirEdgeAttachment

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirEdgeAttachment* `MirEdgeAttachment`

Typedef MirEvent

- Defined in file_include_miral_miral_append_event_filter.h

Typedef Documentation

typedef struct *MirEvent* `MirEvent`

Typedef MirEvent

- Defined in file_include_miral_miral_prepend_event_filter.h

Typedef Documentation

typedef struct *MirEvent* **MirEvent**

Typedef MirFocusMode

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirFocusMode* **MirFocusMode**

Focus mode controls how a surface gains and loses focus.

Typedef MirFormFactor

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirFormFactor* **MirFormFactor**

Form factor associated with a physical output.

Typedef MirInputDeviceCapabilities

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Typedef Documentation

typedef unsigned int **MirInputDeviceCapabilities**

Typedef MirInputDeviceId

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Typedef Documentation

typedef int64_t **MirInputDeviceId**

Typedef MirInputEventModifiers

- Defined in file_include_core_mir_toolkit_events_enums.h

Typedef Documentation

typedef unsigned int **MirInputEventModifiers**

Typedef MirLifecycleState

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirLifecycleState* **MirLifecycleState**

Typedef MirMirrorMode

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirMirrorMode* **MirMirrorMode**

Mirroring axis relative to the "natural" orientation of the display.

Typedef MirNativeBuffer

- Defined in file_include_core_mir_toolkit_mir_native_buffer.h

Typedef Documentation

typedef struct *MirBufferPackage* **MirNativeBuffer**

Typedef miroil::CompositorID

- Defined in file_include_miroil_miroil_surface.h

Typedef Documentation

using `miroil::`**CompositorID** = void const*

Typedef miroil::CreateNamedCursor

- Defined in file_include_miroil_miroil_mir_server_hooks.h

Typedef Documentation

using `miroil::`**CreateNamedCursor** = std::function<std::shared_ptr<mir::graphics::CursorImage>(std::string const &name)>

Typedef miroil::OutputId

- Defined in file_include_miroil_miroil_display_id.h

Typedef Documentation

using `miroil::`**OutputId** = mir::*IntWrapper*<mir::graphics::detail::GraphicsConfOutputIdTag>

Typedef MirOrientation

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirOrientation* **MirOrientation**

Direction relative to the "natural" orientation of the display.

Typedef MirOrientationMode

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirOrientationMode* **MirOrientationMode**

Typedef MirOutputGammaSupported

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirOutputGammaSupported* **MirOutputGammaSupported**

Supports gamma correction.

Typedef MirOutputType

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirOutputType* **MirOutputType**

Typedef MirPixelFormat

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirPixelFormat* **MirPixelFormat**

32-bit pixel formats (8888): The order of components in the enum matches the order of the components as they would be written in an integer representing a pixel value of that format.

For example; abgr_8888 should be coded as 0xAABBGGRR, which will end up as R,G,B,A in memory on a little endian system, and as A,B,G,R on a big endian system.

24-bit pixel formats (888): These are in literal byte order, regardless of CPU architecture it's always the same. Writing these 3-byte pixels is typically slower than other formats but uses less memory than 32-bit and is endian-independent.

16-bit pixel formats (565/5551/4444): Always interpreted as one 16-bit integer per pixel with components in high-to-low bit order following the format name. These are the fastest formats, however colour quality is visibly lower.

Typedef MirPlacementGravity

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirPlacementGravity* **MirPlacementGravity**

Reference point for aligning a surface relative to a rectangle.

Each element (surface and rectangle) has a MirPlacementGravity assigned.

Typedef MirPlacementHints

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirPlacementHints* **MirPlacementHints**

Positioning hints for aligning a window relative to a rectangle.

These hints determine how the window should be positioned in the case that the surface would fall off-screen if placed in its ideal position.

For example, `mir_placement_hints_flip_x` will invert the x component of `aux_rect_placement_offset` and replace `mir_placement_gravity_northwest` with `mir_placement_gravity_northeast` and vice versa if the window extends beyond the left or right edges of the monitor.

If `mir_placement_hints_slide_x` is set, the window can be shifted horizontally to fit on-screen.

If `mir_placement_hints_resize_x` is set, the window can be shrunken horizontally to fit.

If `mir_placement_hints_antipodes` is set then the rect gravity may be substituted with the opposite corner (e.g. `mir_placement_gravity_northeast` to `mir_placement_gravity_southwest`) in combination with other options.

When multiple flags are set, flipping should take precedence over sliding, which should take precedence over resizing.

Typedef MirPointerAcceleration

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Typedef Documentation

typedef enum *MirPointerAcceleration* **MirPointerAcceleration**

MirPointerAcceleration describes the way pointer movement is filtered:

- mir_pointer_acceleration_none: (acceleration bias + 1.0) is applied as a factor to the current velocity of the pointer. So a bias of 0 to results to no change of velocity.
- mir_pointer_acceleration_adaptive: acceleration bias selects an acceleration function based on the current velocity that usually consists of two linear inclines separated by a plateau.

Typedef MirPointerButtons

- Defined in file_include_core_mir_toolkit_events_enums.h

Typedef Documentation

typedef unsigned int **MirPointerButtons**

Typedef MirPointerConfinementState

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirPointerConfinementState* **MirPointerConfinementState**

Pointer Confinement.

Typedef MirPointerHandedness

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Typedef Documentation

typedef enum *MirPointerHandedness* **MirPointerHandedness**

Typedef MirPowerMode

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirPowerMode* **MirPowerMode**

Typedef MirPromptSession

- Defined in file_include_miroil_miroil_mir_prompt_session.h

Typedef Documentation

typedef struct *MirPromptSession* **MirPromptSession**

Typedef MirPromptSessionState

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirPromptSessionState* **MirPromptSessionState**

Typedef MirResizeEdge

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirResizeEdge* **MirResizeEdge**

Hints for resizing a window.

These values are used to indicate which edge(s) of a surface is being dragged in a resize operation.

Typedef MirShellChrome

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirShellChrome* **MirShellChrome**

Shell chrome.

Typedef MirSubpixelArrangement

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirSubpixelArrangement* **MirSubpixelArrangement**

Physical arrangement of subpixels on the physical output.

This is always relative to the "natural" orientation of the display - mir_orientation_normal.

Typedef MirTouchId

- Defined in file_include_miral_miral_toolkit_event.h

Typedef Documentation

typedef int32_t **MirTouchId**

An identifier for a touch-point.

TouchId's are unique per-gesture. That is to say, once a touch has gone down at time T, no other touch will use that touch's ID until all touches at time T have come up.

Typedef MirTouchpadClickMode

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Typedef Documentation

typedef enum *MirTouchpadClickMode* **MirTouchpadClickMode**

MirTouchpadClickMode configures how the touchpad itself should generate pointer button events.

The available click modes may be active simultaneously.

- mir_touchpad_click_mode_none: no active click mode
- mir_touchpad_click_mode_area_to_click: simulate pointer buttons using click areas on the touchpad
- mir_touchpad_click_mode_finger_count: simulate pointer buttons using the number of fingers down

Typedef MirTouchpadClickModes

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Typedef Documentation

typedef unsigned int **MirTouchpadClickModes**

Typedef MirTouchpadScrollMode

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Typedef Documentation

typedef enum *MirTouchpadScrollMode* **MirTouchpadScrollMode**

MirTouchpadScrollMode configures how the touchpad should generate scroll events.

- mir_touchpad_scroll_mode_none: no scroll
- mir_touchpad_scroll_mode_two_finger_scroll: two finger movement generates generates vertical and horizontal scroll events
- mir_touchpad_scroll_mode_edge_scroll: touch movement at the edge of the touchpad genertes scroll events
- mir_touchpad_scroll_mode_button_down_scroll: movement on the touchpad generates scroll events when a button is held down simultaneously

Typedef MirTouchpadScrollModes

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Typedef Documentation

typedef unsigned int **MirTouchpadScrollModes**

Typedef MirTouchscreenMappingMode

- Defined in file_include_core_mir_toolkit_mir_input_device_types.h

Typedef Documentation

typedef enum *MirTouchscreenMappingMode* **MirTouchscreenMappingMode**

Mapping modes for touchscreen devices.

The mode defines how coordinates from the touchscreen frequently referred to as device coordinates are translated into scene coordinates.

This configuration mode is relevant for different classes of input devices, i.e handheld devices with builtin touchscreens or external graphic tablets or external monitors with touchscreen capabilities.

Typedef MirWindowAttrib

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirWindowAttrib* **MirWindowAttrib**

Attributes of a window that the client and server/shell may wish to get or set over the wire.

Typedef MirWindowFocusState

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirWindowFocusState* **MirWindowFocusState**

Typedef MirWindowState

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirWindowState* **MirWindowState**

Typedef MirWindowType

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirWindowType* **MirWindowType**

Typedef MirWindowVisibility

- Defined in file_include_core_mir_toolkit_common.h

Typedef Documentation

typedef enum *MirWindowVisibility* **MirWindowVisibility**

INDEX

Symbols

D

F

K

M

S

T

W